

TEORÍA

ELECTRÓNICA

INDUSTRIAL

ENVÍO 12

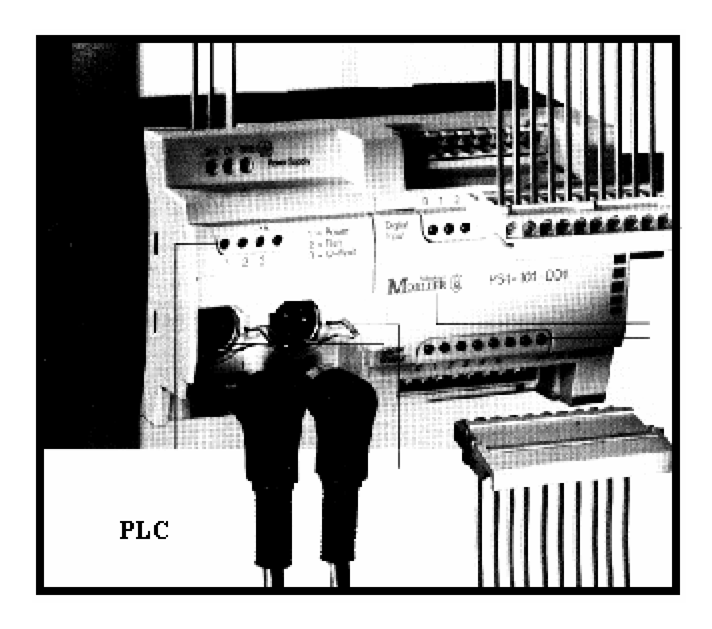


CONTROLADORES LOGICOS PROGRAMABLES (PLC)

INTRODUCCION:

La mayor parte de los procesos de fabricación tienen como finalidad la creación de un producto o la operación de un sistema. Todo esto requiere la ejecución de una secuencia de operaciones, siendo esta particularmente necesaria cuando se realiza la fabricación de piezas discretas.

La secuenciación del proceso se puede realizar manualmente o empleando algún tipo de controlador.



ANTECEDENTES HISTORICOS: Hacia la década de los sesenta todos los procesos de control se encontraban dominados por los relés electromagnéticos, los que en algunos casos formaban verdaderos bancos para realizar una tarea medianamente compleja.

Lo anterior dio origen a una forma de desarrollar sistemas de control que estaban "normados" y existía toda una experiencia teórica y práctica que no era fácil de desechar.

Por requerimientos de la industria y ante los avances que se tenían con elementos de estado sólido (semiconductores) y con el fin de reducir el costo asociado a los sistemas basados en relés, la división **HIDROMATIC** de la **GENERAL MOTORS** identificó e individualizó ciertas características que deberían cumplir él o los componentes que en el futuro reemplazarían a los relés. Estas son:

- **Ser de estado sólido.**
- **Ser flexible como un computador.**
- **Fácil de operar y mantener.**
- **Capaz de operar y resistir ambientes industriales adversos.**

- **Facilidad de programación.**
- **Capacidad de cambiar su aplicación (reorient ar).**

El segundo requerimiento se ubica en el concepto general de lo que era un computador en esa época.

EL CONTROL INDUSTRIAL EN LA ACTUALIDAD:

Con la aparición del microprocesador y la dinámica tecnológica que estos han imprimido a todo lo referente con el control, significó crear un dispositivo de control con un potencial enorme en sus aplicaciones.

La aparición de los controladores lógicos programados **(PLC)**, ha significado una revolución de los procesos de control.

El **PLC** es esencialmente un conductor de eventos en lo convencional. Si un evento ocurre, ciertas acciones se deben llevar a cabo.

Si consideramos al **PLC** como un controlador industrial encargado de monitorear continuamente el estado de las variables en máquinas y equipos de un proce so industrial, este monitoreo dará origen a operaciones y decisiones lógicas.

El análisis que realiza de los estados para tomar una decisión esta estructurado mediante un programa "Booleano" de control, lo que origina acciones **"ON" "OFF"** sobre el estado de las salidas.

Desde esta perpectiva inicial, se podría definir al **PLC** como un control **"ON"** "**OFF**" multivariado.

Los primeros **PLC** tuvieron control **"ON" - "OFF"** y su aplicación se vio limitada a procesos de tipo repetitivo, tales como:

- **Correas transportadoras**
- **Procesos de molienda.**
- **Control de motobombas.**
- **Etc**.

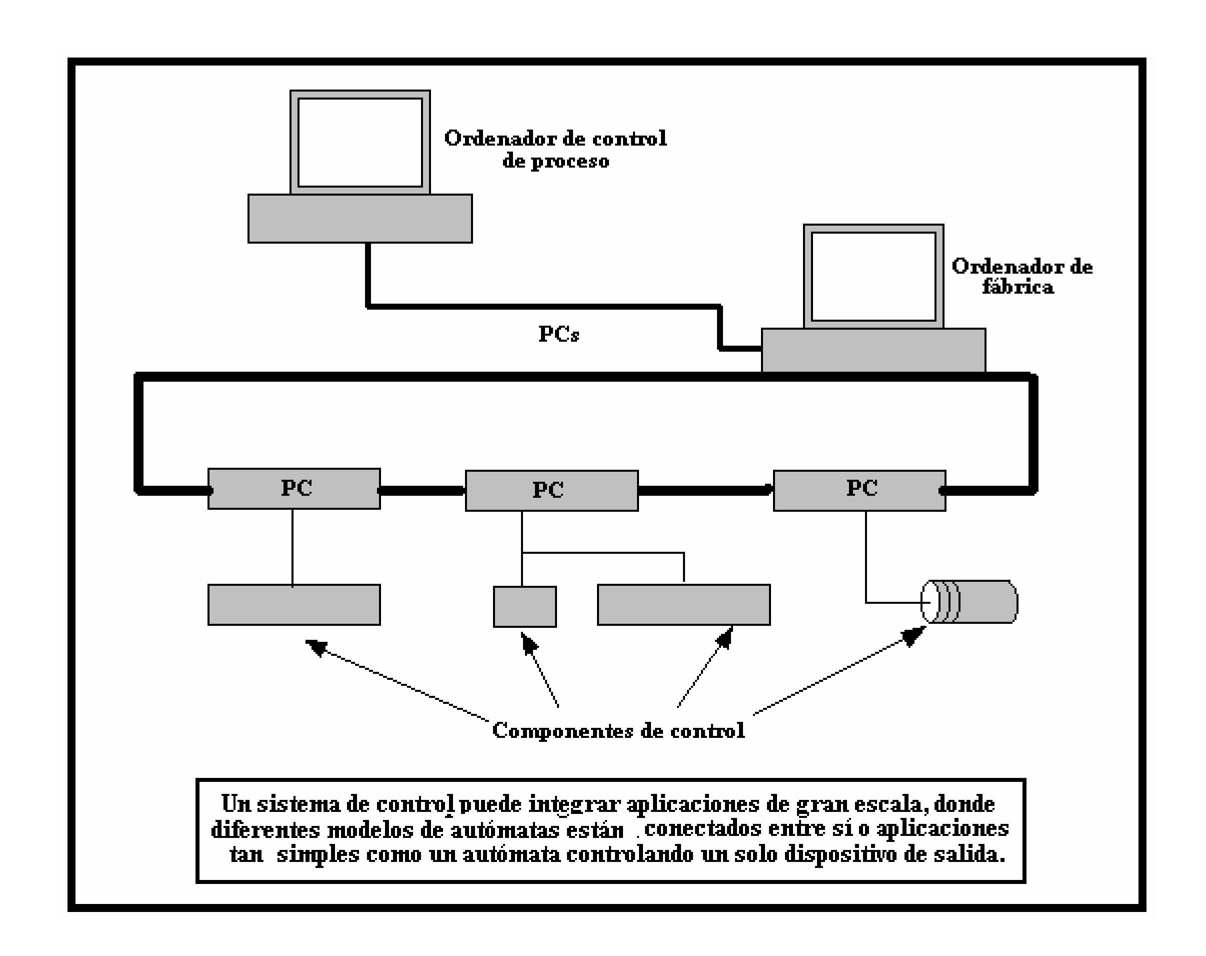
Estos primeros procesos controlados tenían grandes mejoras en relación al proceso basado en relés. Estas mejoras fueron las siguientes:

- **Fácil de instalar.**
- **Pequeño volumen.**
- **Escaso consumo.**
- **Control con indicaciones de diagnóstico**
- **Reorientable al finalizar su aplicación.**

TABLA COMPARATIVA ENTRE UN PLC Y UN RELÉ

CARACTERÍSTICAS	**RELE**	**PLC**
a)Funciones	Solo un gran número de relés permite un sistema complejo	Permite cualquier grado de complejidad
b)Flexibilidad	No, el alambrado debe ser cambiado	Si, es libre, basta con cambiar el programa.
c)Confiabilidad	No, sujeto a falla	Si, pues solo emplea semiconductores
d)Adaptabilidad	No, una vez armado no se emplear en otra aplicación	Si, se adapta a todo, solo cambia el programa.
e)Expandibilidad	No, es difícil su expanción	Si, se adapta a todo, solo cambia el programa.
f)Mantenimiento	Si, requiere mantención periódica	No, solo cambio de partes dañadas.
g)Tamaño	Normalmente grande	Reducidos
h)Diseño	Complejo	Simple.
i)Consumos	Excesivo y calentamientos	Bajo, de 20 a 60 (W)

SISTEMA DE CONTROL



Un sistema de control es un conjunto de dispositivos electrónicos necesarios para controlar un proceso específico. Un sistema de control puede incluir desde un ordenador central hasta los elementos que suministran las entradas y ejecutan las salidas: los interruptores, motores paso a paso, solenoides, sensores.

FUNCIONES DEL PLC:

Se define a un **PLC** como una máquina electrónica diseñada para controlar en tiempo real procesos industriales del tipo secuencial.

Es una "caja negra" donde existen terminales de entrada a los que se conectan:

- **Pulsadores.**
- **finales de carrera.**
- **fotoceldas.**
- **detectores.**
- **etc.**

También existirán salidas a las que se conectarán:

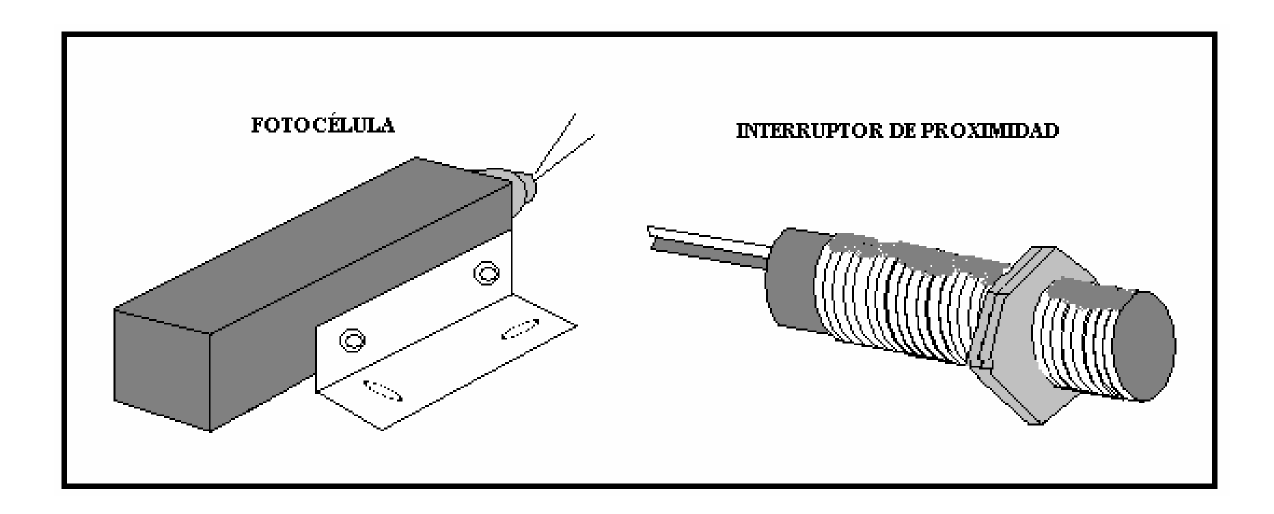
- **Contactores,**
- **electroválvulas.**
- **lámparas,**
- **etc.**

El controlador es el dispositivo del sistema de control que controla directamente el proceso de fabricación, de acuerdo con el programa almacenado en memoria, el controlador recibe los datos de los dispositivos de entrada conectados a él, y utiliza estos datos para monitorear el sistema controlado. Cuando el p rograma ordena tomar alguna acción, el controlador envía las señales correspondientes a los dispositivos de actuación conectados a sus salidas. El controlador se puede utilizar para controlar un proceso simple, repetitivo, o puede conectarse a otros contro ladores o a un ordenador para integrar el control de un sistema complejo.

DISPOSITIVOS DE ENTRADA:

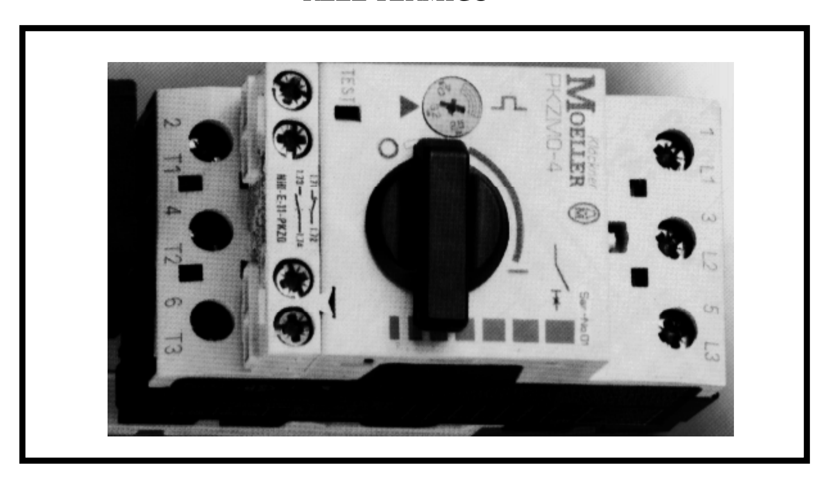
Los controladores pueden recibir señales de entrada provenientes de dispositivos automáticos, tales como :

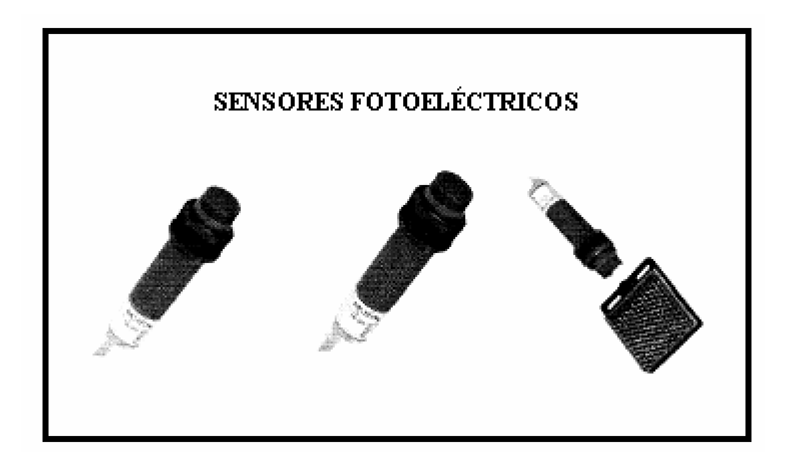
Temporizadores, fotoceldas, sen sores de movimiento, limites de carrera, interruptores de proximidad, etc.





RELÉ TÉRMICO

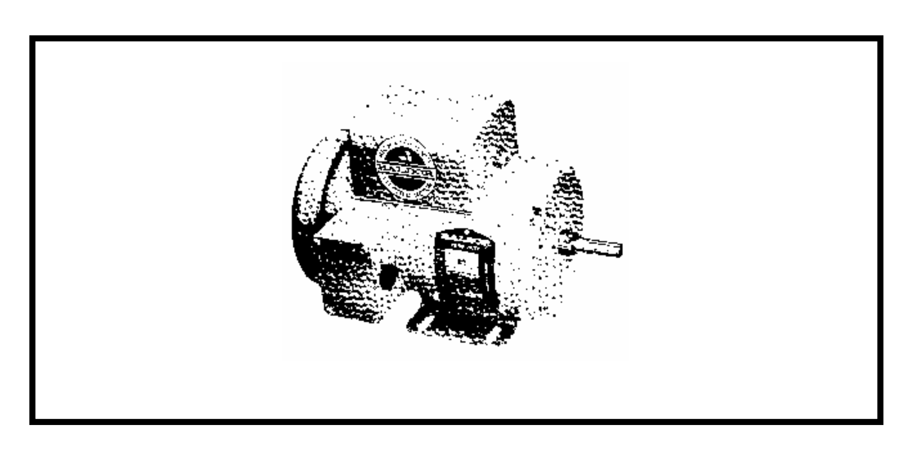


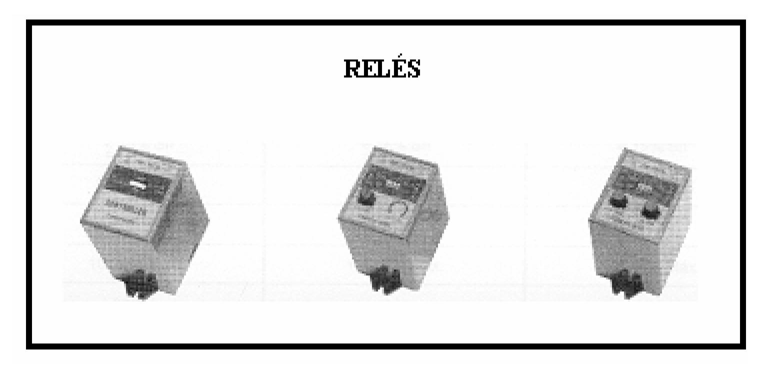


DISPOSITIVOS DE SALIDA:

El controlador puede enviar la salida a un gran número de dispositivos utilizados en automatización.

Prácticamente todo lo imaginable puede ser controlado a través de un **PLC**. Algunos de los dispositivos más comunes son **motores, solenoides, servomotores, motores paso a paso, válvulas, interruptores, indicadores y alarmas.** Ciertos dispositivos como motores, válvulas, solenoides, afectan directamente al sistema controlado; otros como indicadores luminosos o sonoros, y alarmas, sirven como sistemas de monitorización y aviso.





ESPECTATIVAS DEL USO DEL PLC:

Las espectativas iniciales se sobrepasaron con creces, en tal forma que su uso se extendió a otras aplicaciones.

El empleo de microprocesadores les otorgó una flexibilidad y capacidad tal, que en la actualidad realizan funciones tales como:

- **Operaciones aritméticas.**
- **Manejo de datos.**
- **Comunicación entre equipos.**
- **Facilidad para desarrollar nuevos programas** .

El uso de los microprocesadores es, en la actualidad, una alternativa obligada para los controladores industriales, resultando natural que los **PLC** posean características especiales frente a los sistemas de control por relés.

La lógica de relés presenta grandes inconvenientes, pues es:

- **Dificil el análisis y diagnostico de fallas.**
- **Dificil de modificar**.

Los sistemas de control por relés, debido a su sistema de conexionado, de bobinas y contactos, monitorean y actúan sobre las variables en forma paralela.

En cambio un programa almacenado ejecuta una sola instrucción a la vez, lo que impone a los **PLC** limitaciones de paralelismo.

El buen manejo y el conocimiento que existe de la lógica de relés, impuso un lenguaje para aplicar en los controladores lógicos. Lo anterior permite trabajar con un entrenamiento previo en la programación de los **PLC**, aplicando lo ya conocido en un dispositivo diferente a lo habitual, por lo que este elemento se debe manejar como una herramienta más, conociendo toda su potencialidad.

FUNCIONES QUE REALIZA UN PLC:

Todo **PLC** realiza funciones lógicas:

- **a)Serie.**
- **b)Paralelo.**
- **c)Mixtas.**
- **d)Temporizadas.**
- **e)Conteos.**
- **f)Regulaciones, etc**.

El campo de acción debido a las características especiales de los **PLC** es muy extenso. La constante evolución del hardware y software amplia continuamente su campo para satisfacer las necesidades industriales.

Su utilización es fundamental en instalaciones donde es necesario realizar procesos de maniobra, control, señalización; abarcando procesos industriales de cualquier tipo.

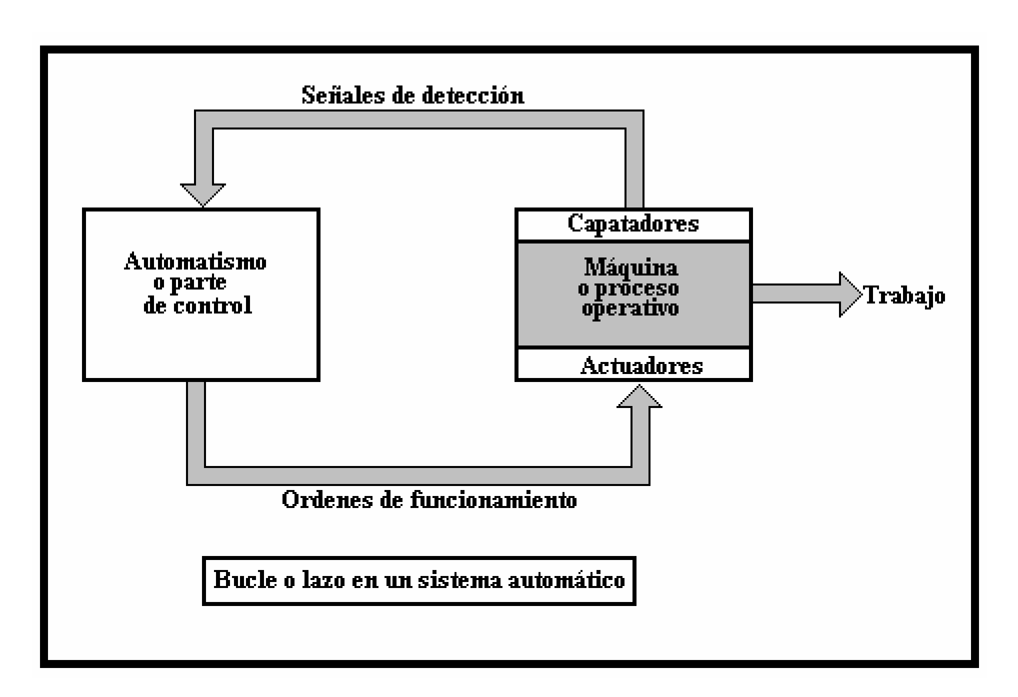
Sus reducidas dimensiones , la facilidad de montaje, el almacenamiento de programas, la rápida utilización hacen que sea de una eficacia en orme.

Su manejo puede ser realizado por personal técnico sin conocimientos de informática.

La tarea del usuario se reduce a realizar un **“ programa”** que no es otra cosa que la relación entre las señales de entrada que se deben cumplir para activar cada sal ida.

PRINCIPIOS DE UN SISTEMA AUTOMÁTICO

Todo sistema de control, por simple que este sea, se basa en el concepto de lazo de control, tal como se muestra en la figura:



Bucle o lazo en un sistema automático

En un sistema de control como el de la figura, resulta conveniente presentar las ventajas y desventajas que presentaría un **PLC**, con respecto a los sistemas tradicionales:

VENTAJAS DE LOS PLC:

- **No es necesario dibujar los esquemas de contacto.**
- **No es necesario simplificar ecuaciónes lógicas dada su gran capac idad de**
- **memoria.**
- **La cantidad de materiales es reducidicima.**
- **Posibilidad de cambios sin cablear de nuevo.**
- **Mínimo espacio.**
- **Menos costo de mano de obra por instalación.**
- **Economía en el mantenimiento.**

- **Posibilidad de controlar varias máquinas con un PLC .**
- **Rapidez para puesta en marcha.**
- **Reorientación si la máquina se elimina.**

DESVENTAJAS DE LOS PLC:

- **Es necesario adiestrar a los técnicos.**
- **Costo inicial.**
- **Es preciso que el proyectista lo conozca tanto en su amplitud como en sus**
- **limitaciones**.

ESTRUCTURA EXTERNA DE UN PLC

El término estructura externa o configuración externa de un autómata programable se refiere al aspecto físico exterior del mismo, bloques o elementos en que está dividido, etc. Desde su nacimiento y hasta nuestros días han sido vari as las estructuras y configuraciones que han salido al mercado condicionadas no solo por el fabricante del mismo, sino por la tendencia existente en el área al que perteneciese: europea o norteamericana. Actualmente, son dos las estructuras más significati vas que existen en el mercado:

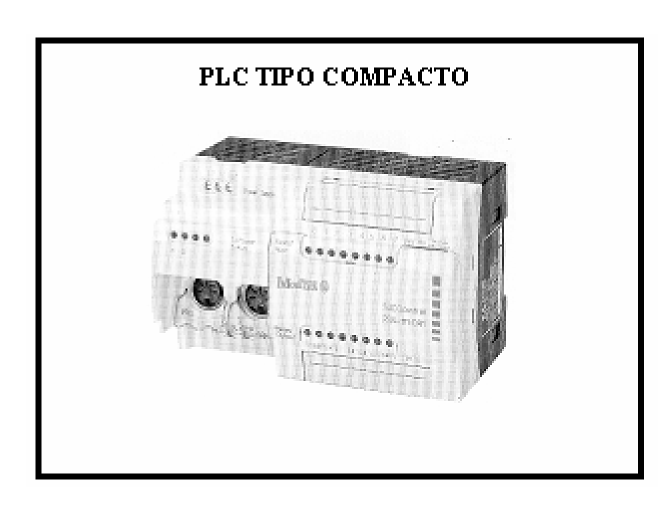
- **Estructura compacta.**
- **Estructura modular.**

Las diferencias significativas entre ambas hacen que las analicemos por separado en los apartados siguientes:

a) PLC DE TIPO COMPACTO.

Este tipo de autómatas se distingue por p resentar en un solo bloque todos sus elementos, esto es, fuente de alimentación, CPU, memorias, entradas/salidas, etc. En cuanto a su unidad de programación, existen tres versiones: unidad fija o enchufable directamente en el autómata; enchufable mediante cable y conector, o la posibilidad de

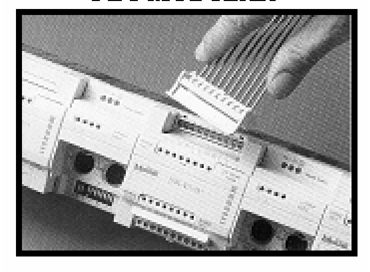
ambas conexiones. Si la unidad de programación es sustituida por un **PC**, nos encontraremos que la posibilidad de conexión del mismo será mediante cable y conector: El montaje del autómata al armario que ha de contenerlo se realiza por cualquiera de los sistemas conocidos: carril DIN, placa perforada, etc.
La figura siguiente nos ilustra sobre este tipo de estructura.



b) PLC DE TIPO MODULAR.

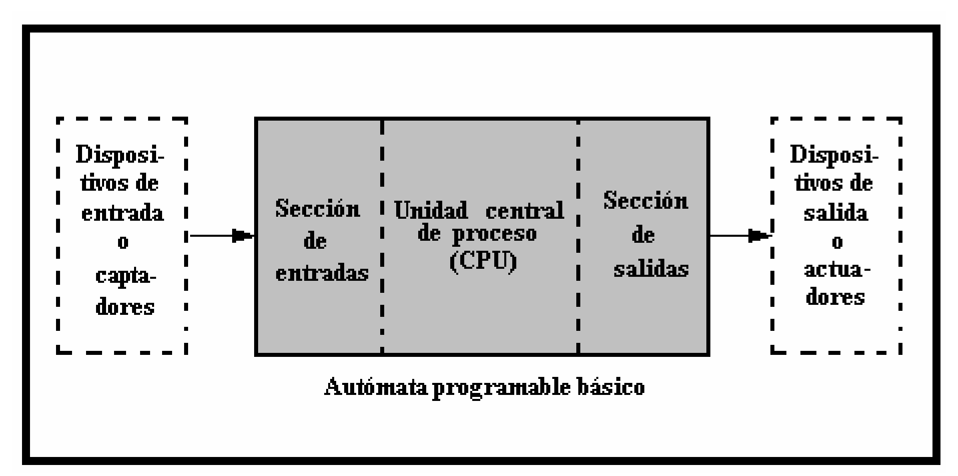
Los **PLC** de tipo modular, como su nombre lo dice, están formados por módulos o partes del mismo que realizan funciones específicas.
Esta forma, a su vez, se puede dividir en dos partes:

- **Estructura tipo Americana**: En este caso se separan las entradas y salidas del resto del **PLC**.
- **Estructura tipo Europea**: En este tipo se separa cada función en un modulo



ESTRUCTURA INTERNA BÁSICA DE UN PLC

En este aparato vamos a estudiar la estructura interna del autómata, o sea, las partes en que se ordena su conjunto físico o hardware y las funciones y funcionamiento de cada una de ellas.
Los autómatas programables se componen esencialmente de tres bloques, tal y como se representa a continuación:



- **La sección de entradas.**

- **La unidad central de procesos o CPU.**

- **La sección de salidas.**

La configuración básica presenta la forma de operar de un **PLC**. Recibe la información desde **sensores** ubicados en el proceso y recibe información de los dispositivos activados. La información se procesa en la **CPU** para entregar como resultado una acción de control.

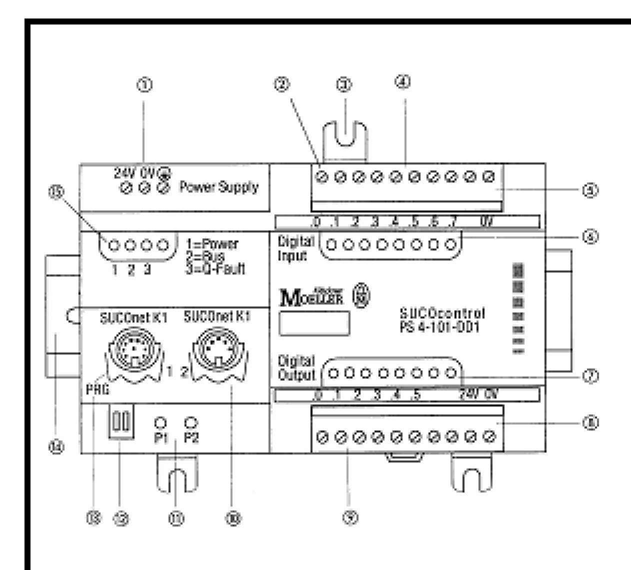
Lo anterior se consigue gracias a un programa **(software)** ingresado por medio de la consola de programación.

a)La sección de entradas. Mediante el interfaz, adapta y codifica, de forma comprensible por la **CPU**, las señales procedentes de los dispositivos de entrada o captadores, esto es, pulsadores, interruptores limites de carrera **, sensores**, etc.;también tiene una misión de protección de los circuitos electrónicos internos del autómata, realizando una separación eléctrica entre éstos y los captadores.

b)La unidad central de proceso (CPU). Es, por así decirlo, la inteligencia del sistema, ya que mediante la interpretación de las instrucciones del programa de usuario y en función de los valores de las entradas, activa las salidas deseadas.

c)La sección de salidas. Mediante el interfaz, trabaja de forma inversa a la de entradas, es decir, decodifica las señales procedentes de la CPU, las amplifica y manda con ellas los dispositivos de salida o actuadores, como lámparas, relés, contactores, arrancadores,

electroválvulas, etc., aquí también existen unos interfaces de adaptación a las salidas y de protección e circuitos internos.

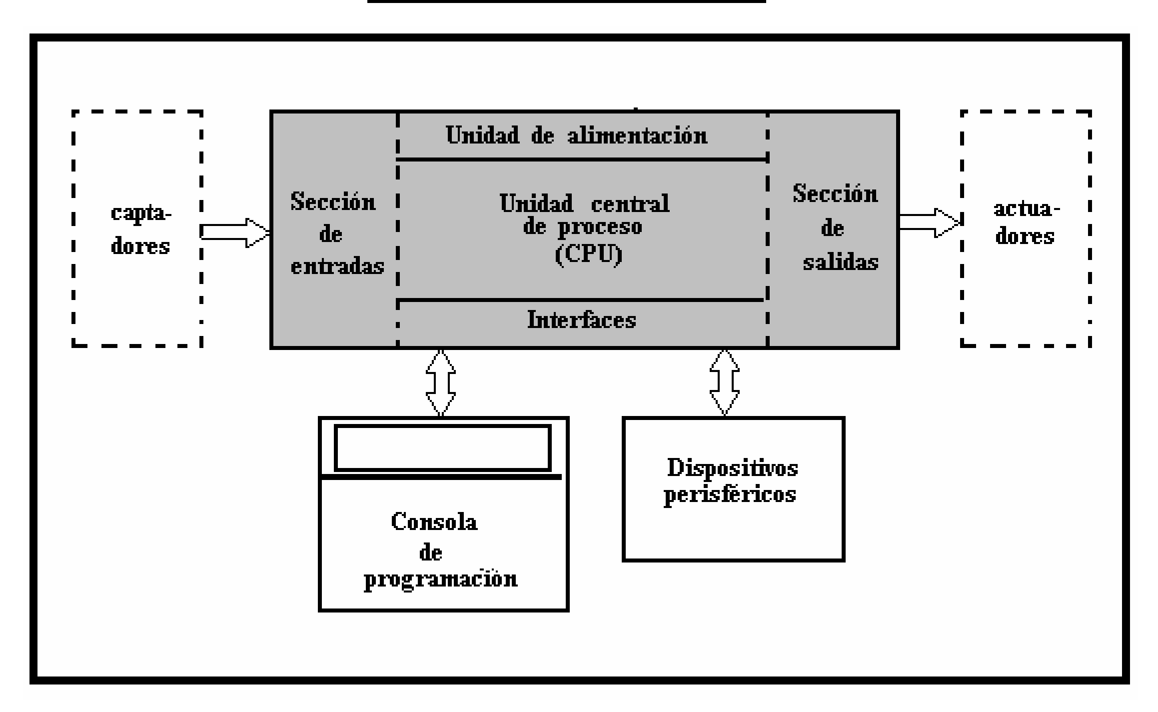


1.-Alimentación (PS4-111 - 230VAC)
2.-Entrada del contactor 3KHz.
3.-Montaje con clip de fijación.
4.-Ocho entradas digitales 24V DC.
5.-Terminales enroscadas enchufables para las entradas.
6.-Indicación del estado de las entradas.
7.-Indicación del estado de las salidas.
8.-Terminal enroscada enchufable para las salidas.
9.-Seis salidas de relé.
10.-Interface SUCOnet K1.
11.-Generador de valor de referencia P1, P2.
12.-Conexión para la resistencia final de bus.
13.-Interface de programación SUCOnet - K1.
14.-Montaje sobre guía simétrica DIN.
15.-Indicación del estado del PLC.

FUNCIONAMIENTO DEL CONTROLADOR

Los controladores reciben las señales de entrada y generan las señales de salida. Al detectarse cambios en el contenido de las señales el controlador reacciona, según el programa grabado por el usuario, para producir las señales de salida. El controlador ejecuta continuamente el programa para conseguir este control

DIAGRAMA EN BLOQUES



El **PS4-100** es un controlador lógico programable **(PLC)** basado en un microprocesador, que puede aceptar señales de entrada desde dispositivos electromecánicos discretos o analógicos en base a un programa generado por el usuario, controlar a través de sus salidas, dispositivos de actuación electromecánicos discretos o analógicos.

El control se realiza mediante la simulación de reles de control internos (Merquer) y otros elementos como temporizadores, comparadores, contadores, registros de desplazamiento (shift registers) y a través de operaciones lógicas simples o aritméticas .

La interacción entre estos elementos de control se determina mediante un programa de control escrito por el usuario. Este programa está compuesto por un número de instrucciones secuenciales que se almacenan en la memoria del **PLC** y que éste ejecuta una por una.

Una vez ejecutadas todas las instrucciones que se encuentran en la memoria, el microprocesador del **PLC** recomienza con la primera instrucción y repite el ciclo . Esta ejecución cíclica del programa del usuario continúa durante el tiempo que el **PLC** se encuentre en modo **RUN**.

El tiempo que demanda recorrer y ejecutar un programa completo depende del número y tipo de instrucciones utilizadas.

El PS4-100 dispone de dos tipos de memorias:

- **Memoria de acceso al azar (RAM) .**

  Es una memoria de acceso aleatorio o memoria de lectura – escritura. En este tipo de memorias se pueden realizar

los procesos de lectura y escritura por procedimiento eléctrico, pero su información desaparece al faltarle la corriente eléctrica.

- **<u>Memoria de lectura solamente, borrable y programable (EPROM)</u>**

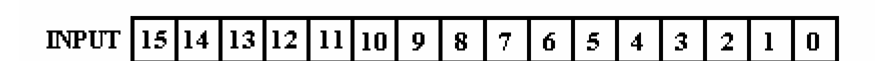
Este tipo de memorias tiene gran aplicación como memorias copia para grabación y archivo del programa del usuario.

Antes de entrar en detalles del **PLC** mismo, es necesario clarificar algunos conceptos en forma previa , conceptos que son fundamentales para lograr una adecuada interpretación del tema al cual nos vamos a referir. Estos conceptos son:

- **Sistema binario**
- **Conversión de binario a decimal**
- **Conversión de decimal a binario**
- **Suma, resta, multiplicación y división de números binarios**
- **Códigos binarios**
- **Sistema hexadecimal**
- **Conversión de hexadecimal a decimal**
- **Conversión de decimal a hexadecimal**
- **Byte**
- **Word**
- **Contadores digitales**
- **Comparadores digitales**
- **Temporizadores**
- **Registros de desplazamiento**

El **PLC** en el cual realizaremos nuestra práctica es el **PS4-100**, el cual es el sucesor del PS3. El PS3 era un PLC que tenía 16 entradas digitales y la misma cantidad de salidas digitales. Además disponía de 4 entradas analógicas y una salida del mismo tipo.

Estas 16 entradas y 16 salidas digitales estaban ubicadas; entradas por la parte superior y salidas por la parte inferior del **PLC**.

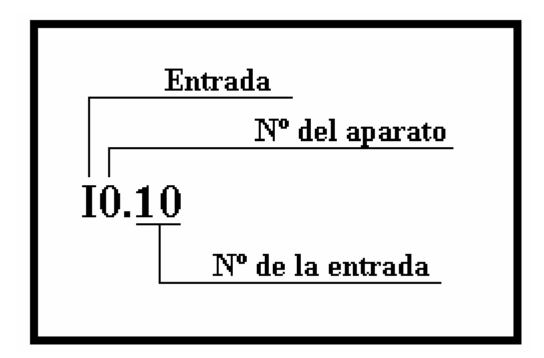


Toda vez que se requería de un mayor número de entradas y salidas , se disponía de otro PS3, con lo cual se lograba duplicar al PS3. A esta nueva puerta de comunicación que se instalaba (PS3) se le denominaba **“ESCLAVO”**, claro está que la máxima cantidad de esclavos que aceptaba el sistema no debía ser superior a 3. El primer PS3 era el maestro y los tres PS3 restantes eran esclavos. Al maestro siempre se le asigna el Nº 0, al primer esclavo el Nº1, al segundo esclavo el Nº2 y al tercer esclavo el Nº3.

- **Maestro = 0**
- **Primer esclavo = 1**
- **Segundo esclavo = 2**
- **Tercer esclavo = 3**

NOTA: normalmente se refieren a ellos por el nombre de **"ESTACIONES"**

Cada PS3 disponía de 16 entradas **(input)** digitales, las cuales, por norma, son designadas como **"I",** y se refieren a ella como **"número de bit",** por lo tanto, si queríamos acceder a la entrada Nº10, debíamos aplicar el siguiente operando:



Donde:

I : Es el código del operando, el cual nos indica que queremos acceder a una entrada

0 : Es el número de estación y nos indica que esta entrada se ubica en el maestro

10 :Es el número de bit y nos indica que queremos acceder a la entrada Nº10.

De acuerdo a esto, el operando indica con qué se ha de realizar una operación. El operando consta de un código de operando (+ extensión) y del parámetro del operando.

El parámetro consta del número de la estación a la cual queremos tener acceso y del número de bit o de entrada de dicha estación.

Así como en el PLC- PS3 existían 16 entradas digitales, también existían 16 salidas (output) digitales, las cuales por norma se designan por la letra "Q", de tal forma que, si en algún momento queríamos acceder a la salida Nº6 de la estación maestro, el operando a través del cual la llamábamos era el siguiente:

Q0.6

En cambio, si queríamos acceder a la salida digital Nº7 de la primera estación esclavo, el operando de llamada era:

Q1.7

Por lo tanto, la numeración era bastante lógica y simple.

OUTPUT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Las entradas (input) analógicas se denominan "IA" y en el PS3 existían 4, por lo tanto, si queríamos tener acceso a la entrada analógica 0 de la estación maestro , el operando a utilizar era el siguiente:

IA0.0

Esto significa que las entradas analógicas de la estación maestro van desde IA0.0 hasta IA0.3

Las entradas analógicas de la estación esclavo Nº1 son IA1.0, IA1.1, IA1.2, IA1.3. y así sucesivamente con los demás esclavos.

ENTRADAS ANALÓGICAS	IA0.0 IA0.1 IA0.2 IA0.3	(ESTACIÓN MAESTRO)
	IA1.0 IA1.1 IA1.2 IA1.3	(ESTACIÓN ESCLAVO Nº1)
	IA2.0 IA2.1 IA2.2 IA2.3	(ESTACIÓN ESCLAVO Nº2)
	IA3.0 IA3.1 IA3.2 IA3.3	(ESTACIÓN ESCLAVO Nº3)

Como en el PLC-PS3 existía solo una salida analógica, teniamos:

SALIDAS ANALÓGICAS	QA0.0	(ESTACIÓN MAESTRO)
	QA1.0	(ESTACIÓN ESCLAVO Nº1)
	QA2.0	(ESTACIÓN ESCLAVO Nº2)
	QA3.0	(ESTACIÓN ESCLAVO Nº3)

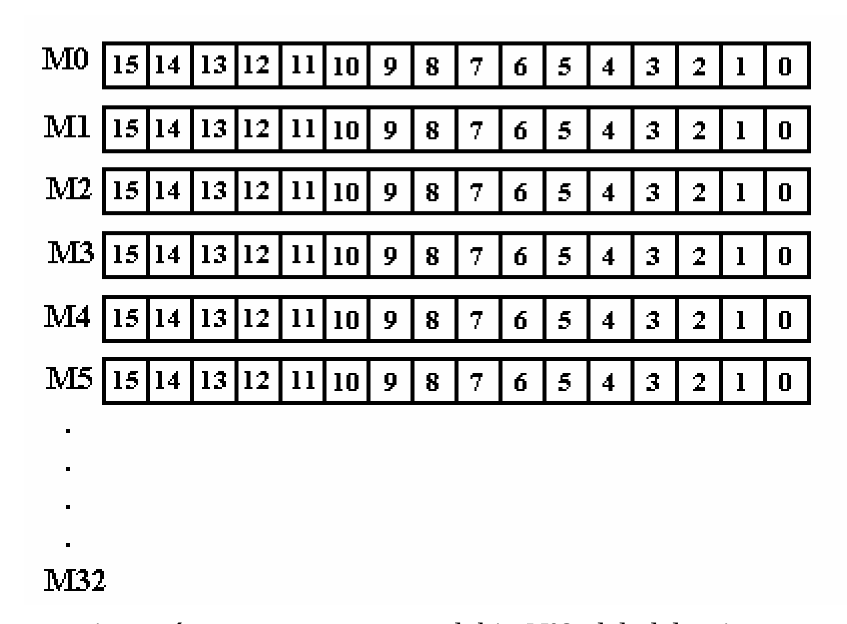
MERQUER

Los merquer del **PLC** son relés internos, relés auxiliares o registros internos de longitud palabra (longitud word), es decir, de 16 bit. Se encuentran normados por la letra "M" y se utilizan para almacenar los resultados provisionales que se producen durante el procesamiento de las señales del PLC. El PLC - PS4 100 dispone de 36 merquer word (MW) utilizables. Estas palabras de merquer pueden direccionarse también en formato de "BIT" o de "BYTE".

En alguna documentación en español, el término **"MERQUER"** es llamado **"BANDERA",** sobre todo en la microelectrónica.

El merquer es un relé auxiliar interno que sustituye a los relés auxiliares utilizados en los esquemas convencionales.

En o PLC - PS3 existían 34 merquer, es decir, desde el merquer 0 (M0) hasta el merquer 33 (M33)



Por lo tanto, si queríamos tener acceso al bit Nº8 del del primer merquer, el operando a utilizar era el siguiente:

M0.8.

Si queríamos leer el bit Nº 3 del noveno merquer, el operando utilizado era:

M10.3

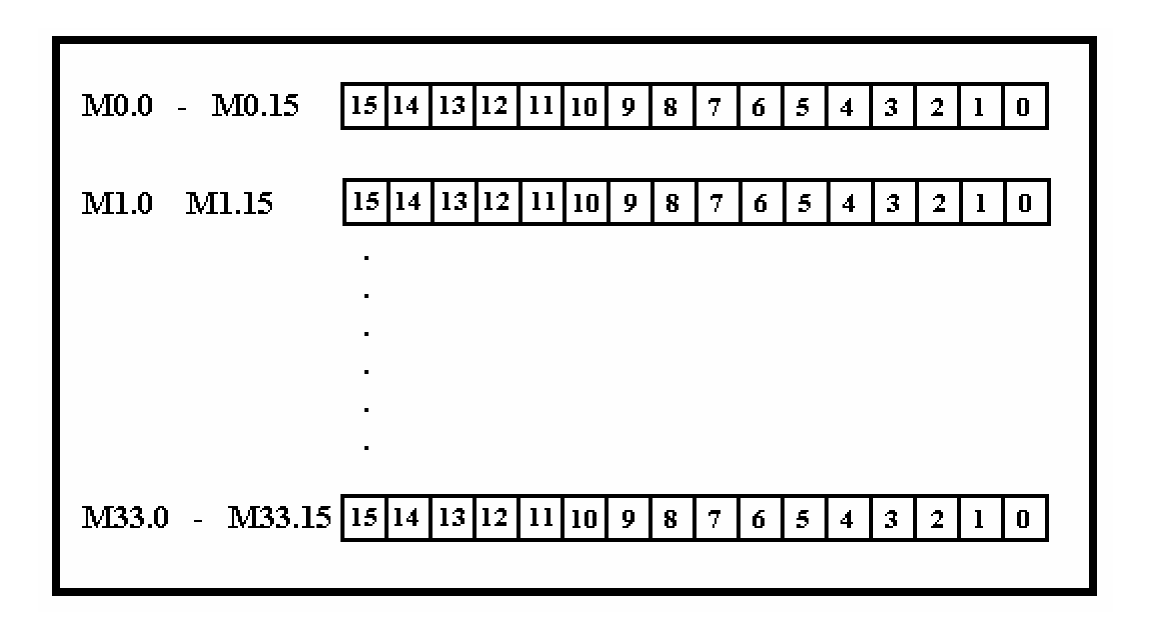
En nuestro programa nosotros mencionamos al antiguo **PLC - PS3**, porque el **PLC - PS4 100** tiene exactamente la misma **CPU** que el viejo **PS3**, incluso el software de programación también es el mismo, lo que significa que con él podemos trabajar tanto en el antiguo **PS3** como en el moderno **PS4**.

La numeración de los elementos del **PS4** es exactamente la misma del **PS3**. Lo que cambia es que en el **PS4** disponemos de 8 entradas digitales y 6 salidas digitales, las cuales se acceden tal como en el viejo **PS3**.

Por ejemplo, si queremos tener acceso a las entradas digitales o a las salidas digitales de la estación maestra 0, se deben ocupar los siguientes operandos:

Entradas digitales	**I0.0 I0.1 I0.2 I0.3 I0.4 I0.5 I0.6 I0.7**	**Estación maestro (0)**
Salidas digitales	**Q0.0 Q0.1 Q0.2 Q0.3 Q0.4 Q0.5**	**Estación maestro (0)**

En nuestro **PLC - PS4 100** tenemos 36 merquer, es decir, desde el merquer **M0.0** hasta el merquer **M35.15**.
No debemos olvidar que los merquer son registros de 16 bit, es decir, de 0 a 15.



Al igual como en el antiguo PS3, a través de la puerta de comunicación podemos ampliar con un máximo de 3 estaciones de esclavos, pero con la diferencia de que ahora tenemos varios diferentes esclavos. Tenemos esclavos que operan solamente con entradas digitales, otros que son solo salidas digitales y otros que son entradas y salidas digitales.

También tenemos esclavos que son entradas y salidas analógicas, por lo tanto los módulos han ampliado mucho en su forma y cantidad y por lo tanto podemos hacer una configuración mucho más acorde a cada aplicación, pero , como la CPU sigue siendo la misma, la forma de llamar a los elementos sigue siendo la misma del PS3.

El PLC - PS4 viene en dos modelos:

Uno de los modelos es el PS4 101 y su alimentación es de 24(v).

El otro modelo es el cual vamos a utilizar para nuestro trabajo; el PS4 111, en el cual la alimentación es de 115 a 230(v), por lo tanto, lo podemos conectar directamente a la red sin problemas.

En la parte superior del PS4 111 se conectan las 8 entradas. A estas entradas se les ha incorporado un simulador. Nosotros a través de este simulador podemos simular, por ejemplo, las botoneras de entrada de nuestro circuito.

En la parte inferior del PS4 111 encontramos 6 salidas digitales (una entrada común y dos salidas, una entrada común y dos salidas, una entrada común y una salida). Dos de ellas están agrupadas y las otras dos son individuales.

Por otro lado, en la parte frontal encontramos dos potenciómetros, que son en realidad las entradas analógicas del PS4 111; solo que no se puede conectar un sensor

externo a ellas, debido a que ya están cableadas internamente a los potenciómetros. Más adelante vamos a utilizar estos potenciómetros para **"SET POINT"**

Esto se utiliza para asignar valores de referencia, por ejemplo, para variar el tiempo de un temporizador.

Los temporizadores internos pueden ser programados en tiempo, pero si les damos un tiempo, por ejemplo, 17,5 segundos, este temporizador siempre va a tener este tiempo, salvo que a través del softward de programación necesitemos modificarlo.

Otra forma es que en lugar de poner un valor fijo, le asignemos este potenciómetro y con ello podremos variar el tiempo que hemos asignado al temporizador. De esta forma tendremos un temporizador más largo sin necesidad de entrar en el softward de programación. Por lo tanto, estos dos potenciómetros también son entradas analógicas, las cuales pueden ser llamadas a través de los operandos **IA0.0 e IA0.1.**

El **PS4 111** dispone de 32 temporizadores. Cualquiera de ellos puede ser programado en forma fija o variable.

Es importante destacar que no necesariamente estos potenciómetros tienen que ver con los temporizadores en forma exclusiva, sino que también los podemos conectar a un comparador, los podemos adicionar a una operación matemática, es decir los podemos ocupar para múltiples finalidades, razón por la cual se les llama **SET POINT** (punto de referencia).

Debemos hacer notar que en este caso no hay salidas analógicas.

Durante el procesamiento es necesario a menudo que el **PS4 111** no solo pueda procesar valores variables sino también valores invariables, o sea constantes. Una constante se designa por la letra **"K"** y se usa para programar valores fijos.

Las constantes pueden ser secuenciadas lógicamente pero no pueden ser asignadas. Se utilizan como números decimales. En función del tipo de dato seleccionado, los valores de las constantes se hallan dentro de los siguientes márgenes numéricos:

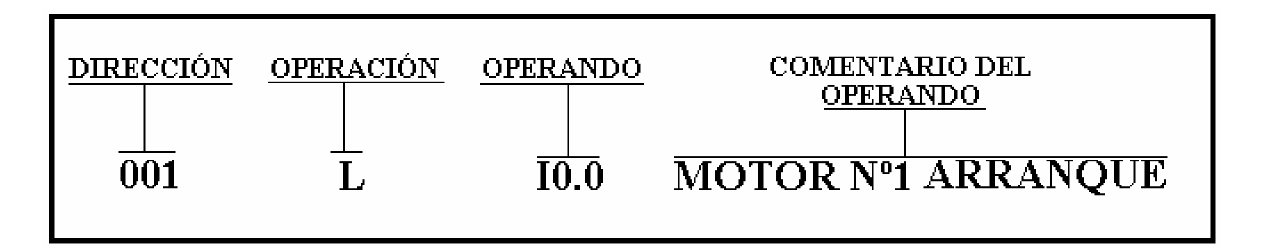
- **BIT:** **K0 y K1**
- **BYTE:** **KB0 a KB255**
- **WORD:** **KW0 a KW65535**

ESTRUCTURA DE UN PROGRAMA

Para la programación de un **PLC** es requisito indispensable definir de forma estructurada el cometido de dicho sistema.

El programa del usuario requiere desde el principio una estructura lógica y clara.

- **INSTRUCCION:** La parte más pequeña de un programa de usuario es la instrucción que se escribe en una línea de instrucción.



Una línea de instrucción consta de varios elementos:

- **DIRECCION**: A todas las instrucciones se les asigna un número consecutivo: la dirección.

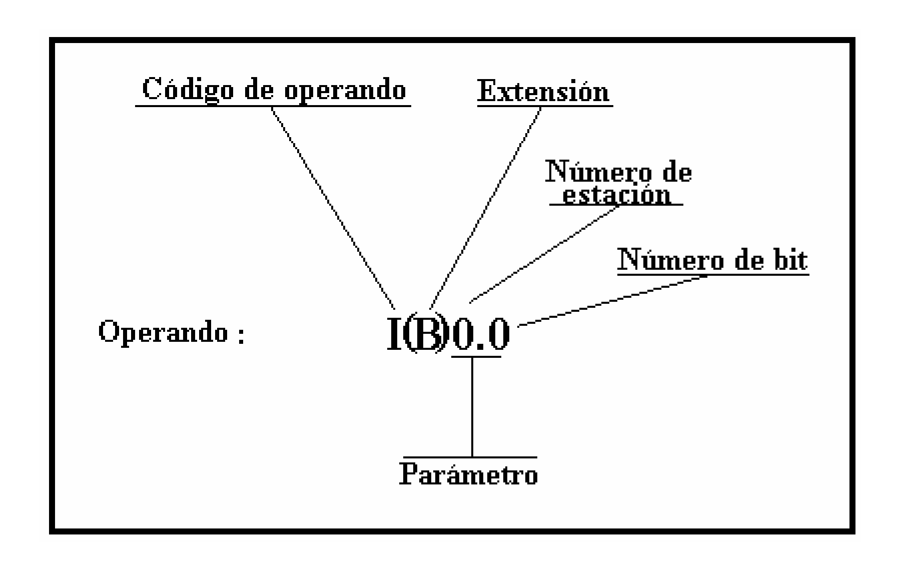
El **PLC** lleva a cabo las instrucciones en el mismo orden en que están las direcciones. Después de la última dirección, el **PLC** empieza de nuevo con la primera instrucción.

- **INSTRUCCION:** La instrucción consta de operación y operando.

- **OPERACION**: La operación indica lo que se ha de hacer.

- **OPERANDO:** El operando indica con qué se ha de realizar una operación. El operando consta de un código de operando (+ extensión) y del parámetro del operando.

- **PARAMETRO**: Consta del número de la estación (Nº de aparato) y del número de bit



La extensión del código del operando indica el tipo de dato utilizado:

- **Sin extensión : Tipo de dato "BIT"**
- **Extensión B : Tipo de dato "BYTE"**
- **Extensión W : Tipo de dato "PALABRA"**

BIT : El bit es la unidad de información más pequeña. Puede tener los estados lógicos **"1" ó "0"** (conectado o desconectado) y sirve para secuenciar rápidamente entradas y salidas.

BYTE : El byte consta de 8 bits y por consiguiente puede tener ocho estados **"1"** ó **"0".** Se utiliza para procesar valores analógicos así como operaciones aritméticas. Un byte representa los valores decimales entre **0 y 255** (dos elevado a ocho).

PALABRA (WORD): La palabra consta de **16 bits ó 2 bytes**. Puede tener 16 veces el estado **"1"** ó **"0".**

Se usa en módulos o como merquer de palabra. Una palabra representa los valores decimales entre **0 y 65535.**

- **COMENTARIO DE OPERANDO:** El comentario de operando debería estar relacionado con el operando mismo y no con la función de la línea de programa. El texto que ha sido escrito después de la primera selección del operando aparece cada vez que éste se entra en cualquier otro lugar

VARIACIONES DE PARAMETROS EN UNA INSTRUCCION TIPO BIT

000 L I 0. 1 = Linea de instrucción (LDI)

DONDE:

000 = Operación (indica lo que se ha de hacer)

L = Dirección (orden en que se llevan a cabo las instrucciones)

I = Código del operando

0 = Estación o unidad (indica si la instrucción se llevará a cabo en el aparato base (maestro) o en una extensión (esclavo)).

1 = Entrada como BIT

Las operaciones pueden ser:

L = CARGAR
A = AND
O = OR
XO = OR EXCLUSIVA
= = ASIGNACIÓN
S = ACTIVAR
R = DESACTIVAR

Los códigos del operando pueden ser:

N = NEGACIÓN
I = ENTRADA
Q = SALIDA
M = MERQUER
K = CONSTANTE

Las unidades I y Q (estación con entradas y salidas) pueden ser:

0 = APARATO BASE O MAESTRO
1 = PRIMERA ESTACIÓN ESCLAVA
2 = SEGUNDA ESTACIÓN ESCLAVA
3 = TERCERA ESTACIÓN ESCLAVA

Con Merquer M : Desde el Merquer 0 hasta el Merquer 35

Con constantes K : 0 ó 1

Entradas I : 0 15
Salidas Q : 0 15
Merquers : 0 15
(No aplicable a constantes)

VARIACIONES DE PARAMETROS EN UNA INSTRUCCION TIPO BYTE

000 OIB0.8	= Linea de instrucción (LDI)
000	= Dirección
0	= Operación
IB	= Código del operando + extensión
O	= Número de estación o unidad
8	= Entrada como BYTE

Las operaciones que se pueden realizar con instrucciones tipo Byte son las siguientes:

L = CARGAR A = AND O = OR XO = OR EXCLUSIVA = = ASIGNACIÓN	
ADD = SUMA SUB = SUSTRACCIÓN MUL = MULTIPLICACIÓN DIV = DIVISIÓN NOT = NEGACIÓN	DESPLAZAMIENTO

Otros códigos de operando aplicables a instrucciones tipo son las siguientes:

IB = ENTRADA COMO BYTE
QB = SALIDA COMO BYTE
MB = MERQUER COMO BYTE
IA = ENTRADA ANALÓGICA (BYTE)
QA = SALIDA ANALÓGICA (BYTE)
KB = CONSTANTE COMO BYTE

0 = UNIDAD O ESTACIÓN MAESTRA
1 = PRIMERA ESTACIÓN O ESCLAVO
2 = SEGUNDA ESTACIÓN O ESCLAVO
3 = TERCERA ESTACIÓN O ESCLAVO

Con Merquer M : Desde el Merquer 0 al Merquer 35
Con constantes KB : 0 255

Entradas IB	= 0 / 8
Salidas QB	= 0 / 8
I analógicas (IA)	= 0 3
Q analógicas (QA)	= 0
Merquers (MB)	= 0 / 8
No aplicables con constantes	

VARIACIONES DE PARAMETROS EN UNA INSTRUCCION TIPO PALABRA (WORD)

010 LIW1	= Linea de instrucción (LDI)
O10	= Dirección
L	= Operación
IW	= Operando
1	= Número de estación

En una instrucción tipo Word las operaciones pueden ser:

L	= Cargar
=	= Asignación

En instrucciones tipo Word los posibles operandos a utilizar son:

IW	=	Entrada como palabra.
QW	=	Salida como palabra.
MW	=	Merquer como palabra
KW	=	Constante como palabra.
0	=	Estación base o maestro.
1	=	Primera extención o esclavo.
2	=	Segunda extención o esclavo.
3	=	Tercera extención o esclavo.
Con Merquer MW: Desde el MW0 hasta el MW35		
Con constantes KW : 0 ....65 535		

SECUENCIA: Una sucesión consecutiva de varias instrucciones que deben cumplir con ciertas condiciones se denomina secuencia. La primera línea de comando de una secuencia ha de contener una instrucción de carga (por ej. LI0.0)
El tipo de dato **(BIT, BYTE o WORD)** del operando de esta instrucción de carga determina el tipo de dato de toda la secuencia. El tipo de dato no puede cambiarse dentro de una misma secuencia.
La secuencia puede concluirse con la siguiente instrucción:

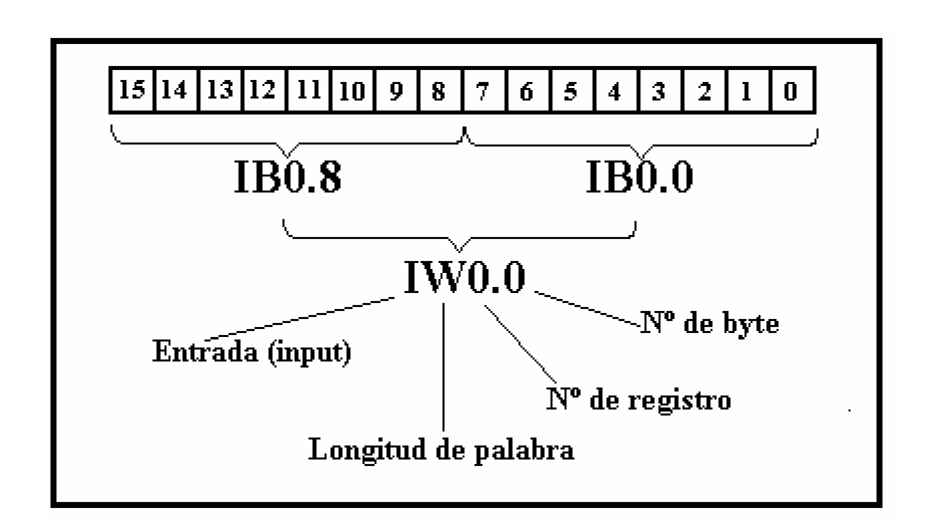
= **Asignación.**

Ejemplo de una secuencia con el tipo de dato "BIT":

LI0.0 **Motor 1 arranque.**
ANI0.2 **Interruptor final de carrera.**

AM2.0 Display motor 1
=Q0.0 Motor 1 en funcionamiento.

Vamos a entrar un momento en la definición de **bit, byte y palabra** , dado que es importante para la comprensión del tema.
Si tenemos un registro, como es el caso del viejo PS3, de 16 entradas.



Si estas son las 16 entradas (del 0 al 15), podemos trabajar bit por bit, es decir, información por información, llamando a cada elemento.
Por ejemplo, si queremos llamar al bit Nº2, este sería: **I0.2.**
Por lo tanto, tanto en el PS3 como en el PS4 tenemos la posibilidad de programar bit por bit, o los 8 bit en forma simultánea, es decir, un byte a la ve z.
Por ejemplo, si queremos llamar los primeros 8 bit, de derecha a izquierda, en forma simultánea, la instrucción será: IB0.0, donde:

I = Entrada.
B = Byte (8 bits).
0 = Registro Nº0
0 = Bit de menor peso

De esta forma, con la instrucción IB0 .0 estaremos llamando en forma simultánea a los primeros 8 bits (del 0 al 7).
También pueden ser llamados los últimos 8 bits (del 8 al 15) en forma simultánea. La instrucción para este cometido será: IB0.8.

- **Para el de menor peso= IB0.0 (Serán leídos los bits del 0 al 7)**
- **Para el de mayor peso= IB0.8 (Serán leídos los bits del 8 al 15)**

Si la instrucción lleva dato tipo byte, solo podrán ser leídos los primeros 8 bits **(0 al 7)** o los últimos 8 bits **(del 8 al 15),** pero nunca los que se encuentren entremedio, como por ejemplo los bits del 4 al 11.
En el PS4 100 tenemos la posibilidad de leer los 16 bits en forma simultánea. La instrucción para esta llamada deberá ser: **IW0 o IW0.0** (el cero final es optativo).
En cuanto a las entradas analógicas que habíamos mencionado, las cuales son los dos potenciómetros vistos anteriormente, tienen una resolución de 8 bits, es decir, si el potenciómetro lo llevamos a un extremo nos da el valor cero (2º) mientras que en el otro extremo nos da el valor 255 (2).Por lo tanto, una entrada analógica es siempre de longitud Byte, por lo tanto, IA0.0 es lo mismo que **IB0.0 o IB0.8**, dado que siempre son 8 bits simultáneos los que leeremos. Por lo tanto, el **PS4 - 100** tiene entradas analógicas de 8 bits y por lo tanto son de longitud Byte.
En cuanto a las constantes (**K**), también podemos usar constantes de Bits, byte o de Word.

Si las constantes son de Bits deberán ser llamadas K (K0 o K1).

Si estamos trabajando en Byte, la s constan tes deberán ser llamadas KB (KB0 KB255).

Si se esta trabajando en Word, las constantes deberan ser llamadas KW (KW0 KW65.535).

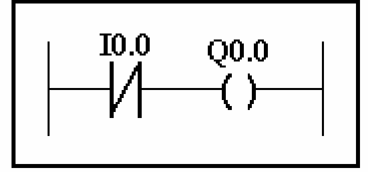
INSTRUCCIONES

LOAD (L): Función: Inicia la operación de cada línea lógica. Cuando una línea comienza con un contacto NA, esta instrucción indica que se comienza en la dirección especificada una línea o sublínea de diagrama de contactos. Utilice esta instrucción para cada línea lógica que comience con un contacto NA..

LI0.0
=Q0.0
EP

I0.0 Q0.0

LOAD - NOT (LN): Función: Inicia la operación de cada línea lógica con un contacto NC. Se utiliza esta instrucción en vez de L, cuando una línea lógica empieza con un contacto NC.



LNI0.0
=Q0.0
EP

AND (A): Función : Realiza la operación lógica AND de dos o más contactos en serie. Esta instrucción realiza la operación lógica AND del resultado de una operación lógica salvada previamente del registro con el relé especi ficado. El resultado de la operación se almacena luego en el registro.

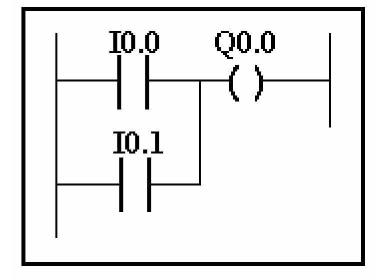
LI0.0
AI0.1
=Q0.0
EP

AND-NOT (AN) Función: Conecta en serie contactos NC. Esta instrucción invierte el contenido del relé especificado y luego realiza la operación lógica AND con los contenidos del registro, quedando el resultado almacenado en el registro.

LNI0.0
ANI0.1
=Q0.0
EP

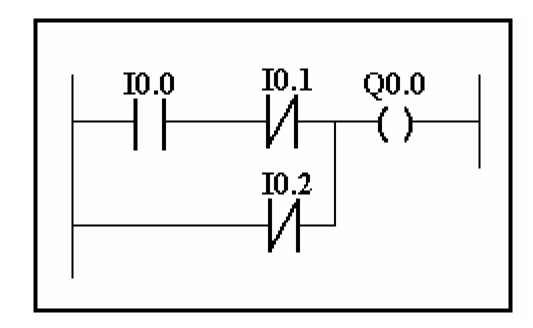
OR (O): Función: Realiza la operación lógica OR entre un relé especificado y el contenido del registro, es decir, conecta dos ó más contactos en paralelo. Esta instrucción realiza la operación lógica OR entre los contenidos del registro con el relé especificado. El resultado lo almacena en el registro.

LI0.0
OI0.1
=Q0.0
EP

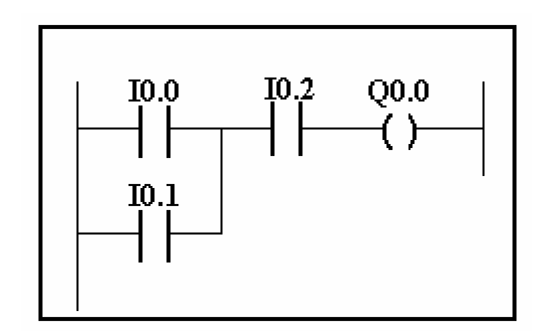


OR-NOT (ON): Función: Conecta en paralelo contactos NC. Esta instrucción realiza la operación lógica OR entre los contenidos del registro y los c onteni dos de un relé especificado. El resultado lo almacena en el registro.

LI0.0
ANI0.1
ONI0.2
=Q0.0
EP
LI0.



Lo más complejo en el manejo de un **PLC** no es la programación propiamente tal, sino que el pasar de lo que requiere el cliente en su aplicación, a un esquema eléctrico. Es justamente en esta parte del trabajo donde se pierden l as horas, puesto que el trabajo de ingeniería previo es el difícil. Por lo tanto, para nuestro estudio de la programación del **PLC**, vamos a suponer que este trabajo ya lo hemos hecho y que ya contamos con los circuitos eléctricos necesarios.
Supongamos el siguiente circuito eléctrico:



El circuito mostrado consta de dos contactos en paralelo con uno en serie que actúa sobre una salida.
A los contactos en paralelo los llamaremos I0.0 e I0.1. El contacto serie se denominará I0.2 y la salida Q0.0.
Nosotros a través de nuestro propio lenguaje diríamos: Si I0.0 o I0.1 y I0.2, entonces Q0.0.
Bueno, si esto lo pasáramos al ingles, el SI, por ser el inicio de la secuencia lo reemplazamos por la operación LOAD (L), el O lo reemplazamos por la operación OR (O), el I lo reemplazamos por la operación AND (A) y el entonces por la operación EQUAL (=).
Por lo tanto, esto sería exactamente lo que hay que programar:

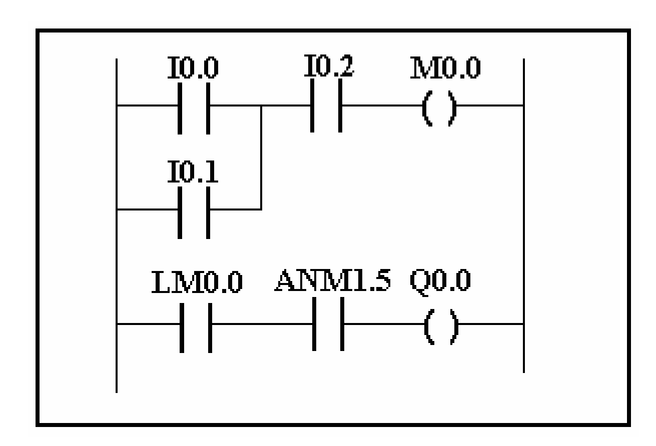
LI0.0
OI0.1
AI0.2
=Q0.0

Con esto ya tenemos una regla:

a)Los contactos en paralelo son programad os con la operación O.
b)Los contactos en serie se programan con la operación A.
c)La asignación de una bobina con el símbolo de la operación =.
d)El inicio de una secuencia es siempre con la operación L .

Por lo tanto, la regla más importante y bás ica ya la tenemos solucionada. Una vez que ya conocemos esto, lo más importante es profundizar las posibles combinaciones. Obviamente que existen nuevas reglas que aprender.

A continuación realizaremos un nuevo circuito:



El circuito mostrado es prácticamente igual al circuito anterior, solo que ahora no actúa sobre la bobina de una salida, sino sobre la salida de un relé interno (Merquer).

Al circuito se le ha incorporado el contacto abierto de un merquer, en serie con el contacto cerrado de otro merquer que sí actúa sobre una salida de bobina.

De acuerdo a esto la secuencia a programar en el PLC será la siguiente:

LI0.0 **Inicio de la primera secuencia**
OI0.1
AI0.2
=M0.0
LM0.0 **Inicio de la segunda secuencia**
ANM1.5
=Q0.0

Para la primera secuencia tendríamos que decir:

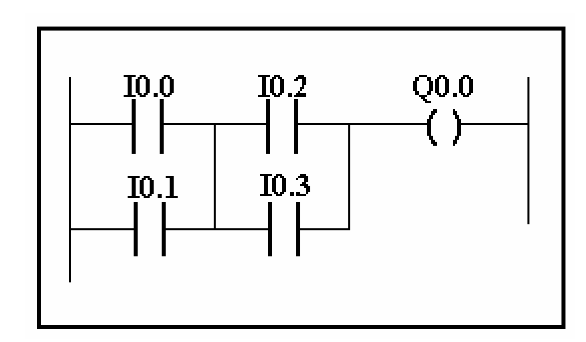
Si I0.0 o OI0.1 y I0.2, entonces M0.0.

Ahora, a continuación, iniciamos una nueva secuencia con LM0.0, que es un contacto abierto. Este contacto abierto se encuentra en serie con un contacto cerrado. Los contactos cerrados, en la informática corresponden a una información negada (A: serie N : negado = AND- NOT = AN), es decir, ANM1.5. Por lo tanto, los contactos cerrados se programan con una N:

AN si es un contacto cerrado en serie.

ON si es un contacto cerrado en paralelo.

El próximo circuito que desarrollaremos se muestra a continuación:



En el circuito tenemos dos contactos N.A.en paralelo y en serie con otros dos contactos N.A. en paralelo.

Para el análisis de este nuevo circuito vamos a entrar en el funcionamiento interno de la **CPU**.

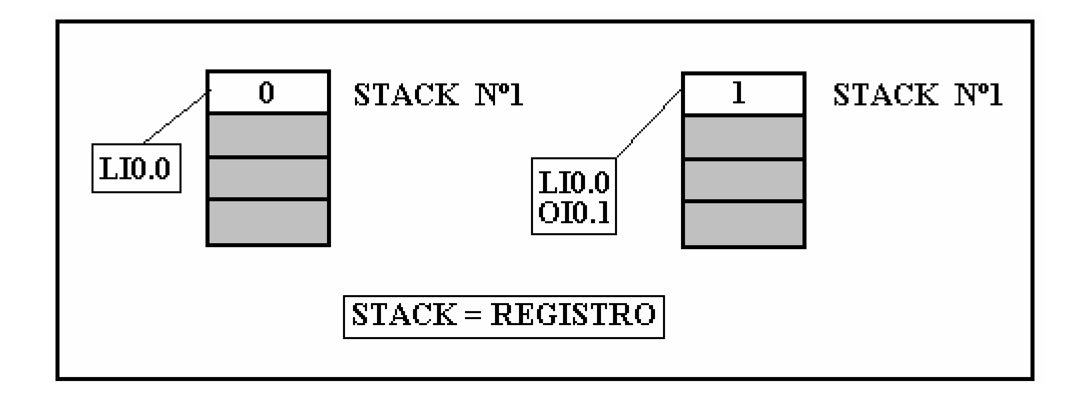
La **CPU** dispone de un registro auxiliar con el cual trabaja. Este registro, en ingles, se denomina **STAC**K.

Veamos por ejemplo que hace la **CPU** con la siguiente secuencia de instrucciones:

LI0.0
OI0.1

Lo primero que hace la **CPU** es leer el estado de la entrada I0.0. En el momento de la lectura I0.0 puede estar abierto (0) o cerrado (1). Si en el momento de la lectura I0.0 se encuentra abierto, equivale a un 0 y por lo tanto, esa información la escribe en el **STACK Nº1**.

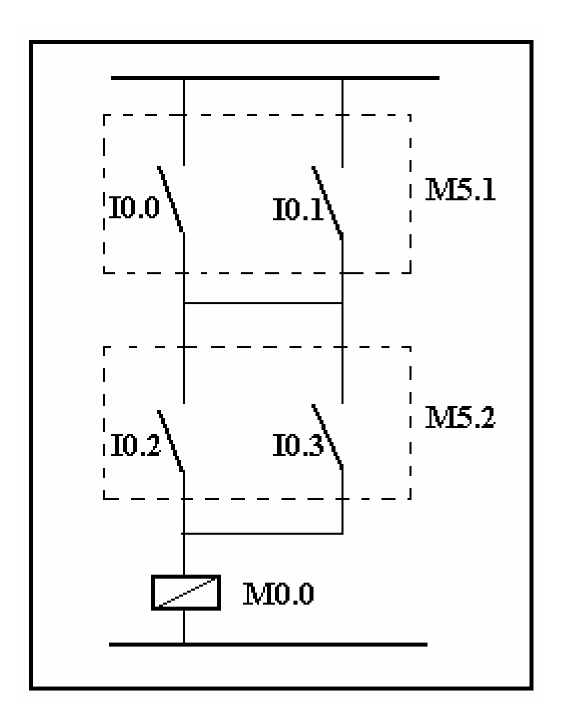
Imaginemos que en ese mismo instante el contacto I0.1 se encuentra cerrado (1). Esto para la **CPU** significa hacer una conexión en paralelo **(OR)** entre el estado del contacto I0.0, archivado en el **STACK Nº1** y el estado del contacto I0.1, por lo tanto, lo que se reescribe en el **STACK Nº1** es el resultado de esta operación OR (0 or 1 = 1).



Existen dos posibilidades para la solución de este ejercicio circuital.:

La primera versión es utilizada por muchos programadores debido a que es la más clara y fácil de entender.

El inconveniente que tiene esta versión es que su desarrollo resulta mucho más largo de ejecutar. Para lograr una adecuada interpretación de como operar con esta primera versión, vamos a cambiar nuestro circuito original a un circuito equivalente.

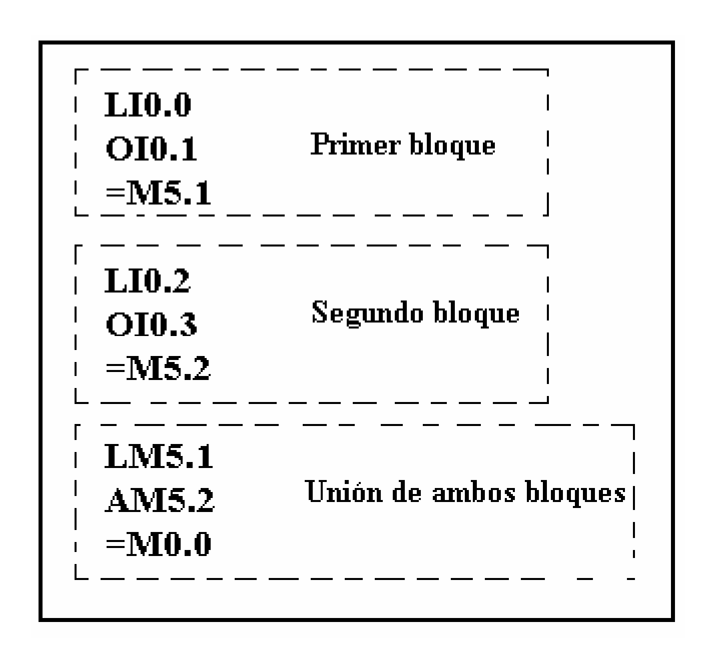


El desarrollo se inicia tomando el primer bloque paralelo y asignarle un merquer, por ejemplo, el merquer **M5.1.**

Al segundo bloque le asigno el merquer **M5.2.**

Luego , se toman se toman estas asignaciones, se unen en serie y se les asigna un nuevo merquer, por ejemplo**, M0.0.**

De acuerdo a esto, la programación sería:



Por lo tanto, nos hemos apoyado de una ayuda, en el sentido de que hemos dividido nuestro esquema eléctrico en bloques y a cada bloque le hemos asignado un merquer, uniendo posteriormente a ambos en un nuevo merquer.

La segunda versión es mucho más confortable, corta y por lo tanto, rápida, pero para comprenderla vamos a volver al funcionamiento in terno de la **CPU**.

Nuestro nuevo programa, de acuerdo a esta segunda versión sería:

LI0.0 **OI0.1**	**Primera secuencia**
LI0.2 **OI0.3**	**Segunda secuencia**
A	**Unión en serie de ambas secuencias**
=M0.0	**Salida**

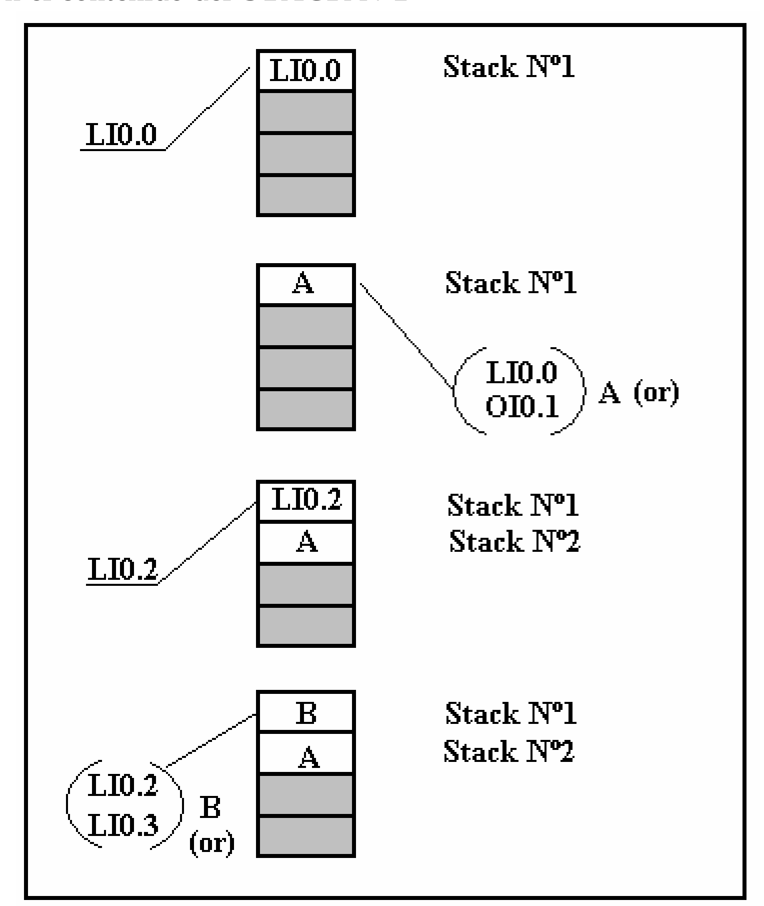
Podemos darnos cuenta de que hemos creado dos inicios de secuencia sin haberlas cerrado. Debemos recordar que toda vez que se inicia una secuencia con la instrucción **LOAD (L),** la misma debe ser cerrada a través de una asignación **(=)**.

Para comprender esto, analizaremos como actúa la **CPU:**

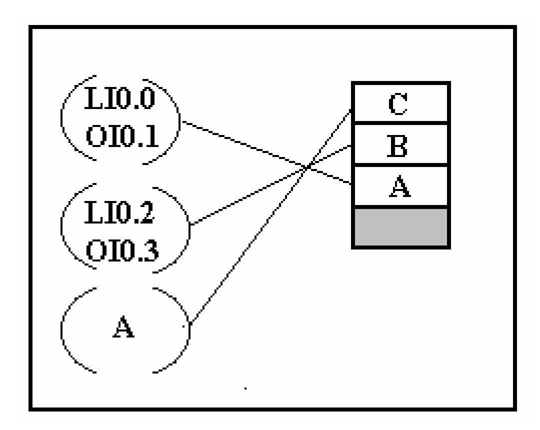
Primer paso: La **CPU** ve en que estado se encuentra el contacto I0.0 y escribe esta información en el **STACK Nª1**.

Segundo paso: La **CPU** mira el estado del contacto I0.1 y lo combina en paralelo (OR) con el estado del contacto I0.0 almacenado en el **STACK Nº1** (S = A o B), sobreescribiendose el resultado en el **STACK Nº1**.

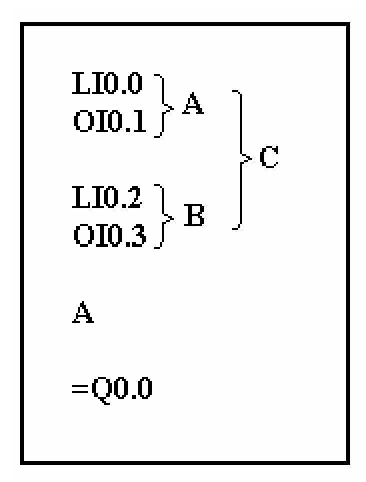
Tercer paso: En la tercera línea se encuentra otro inicio de secuencia **(L),** lo que significa leer el estado de I0.2 y almacenarlo en el **STACK Nº1**, pero no debemos olvidar que en el **STACK Nº1** se encuentra almacenado el resultado de las dos primeras líneas, por lo tanto, este último es desplazado un STACK más abajo, es decir al **STACK Nº2**. Por lo tanto, en el **STACK Nº1** se encuentra almacenada la última información recopilada por la **CPU**. Esto significa que la **CPU** actualiza la información contenida en el **STACK Nº1** y la que había la traslada al **STACK Nº2,** por lo tanto, la **CPU** siempre trabaja sobre el **STACK Nº1**.
Ahora, obviamente, tenemos dos secuencias abiertas y ninguna cerrada. Esto, de paso sea dicho, en informática es trabajar con paréntesis, y claro está, cada vez que se inicia la apertura de un paréntesis, habrá que cerrarlo.
A continuación debemos decirle a la **CPU** que hacer con la información contenida en los STACK, puesto que tiene que haber una conexión en serie del contenido del **STACK Nº2** con el contenido del **STACK Nº1**



Lo que debemos hacer con los bloques **A y B** es conectarlos en serie. Para lograr este fin, usaremos la operación **AND (A)** sin ningún operando. Una operación A sin operando, significa hacer una conexión en serie del contenido del primer ST ACK con el contenido del segundo STACK, es decir una conexión en serie de estos dos bloques, quedando almacenado el resultado en el **STACK Nº1**. De esta forma, el contenido del primer bloque es trasladado al **STACK Nº3**, el contenido del segundo bloque es cambiado al **STACK Nº2** y el resultado de la conexión serie de ambos bloques queda actualizado en el **STACK Nº1.**



Por lo tanto, el programa final es:

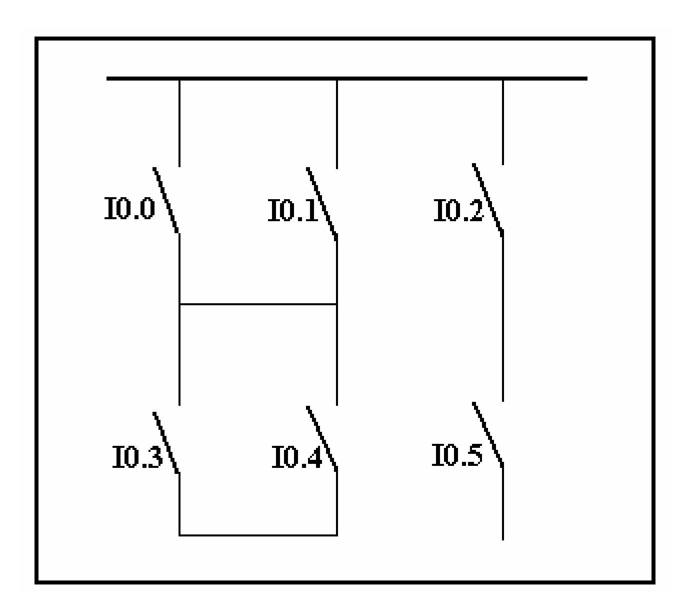


En realidad, a través de un STACK lo que podemos hacer es abrir un paréntesis y co mo se puede observar, tenemos 8 STACK, lo que significa que puedo abrir 8 paréntesis a lo largo de toda mi secuencia. Se supone que al termino de estas secuencias vamos a quedar prácticamente sin paréntesis.

Hemos abierto una secuencia y la hemos cerrado. En cada secuencia tiene que haber un **LOAD** y una asignación, por lo tanto, por cada **LOAD (L)** tiene que haber una asignación. En nuestro caso, orientados hacia el caso analizado, con **L** hemos abierto un paréntesis y con **A** lo hemos cerrado. De acuerdo a esto, la regla de oro es la siguiente:

EN UNA SECUENCIA MUY LARGA, SE DEBEN SUMAR TODAS LAS L, Y TIENEN QUE HABER TANTAS OPERACIONES SIN OPERANDO MENOS UNO.

Para este caso se cumple perfectamente la regla, puesto que existen 2 operaciones **L** con operando y una operación **A** sin operando.
Si existen 7 operaciones **L** , tienen que haber 6 operaciones sin operando, las cuales pueden ser **O** ó **A.**
Veamos otro ejemplo:



En la figura superior mostramos otro ejemplo con operaciones **STACK** en paralelo y en serie.
En el circuito tenemos dos contactos en paralelo que se encuentran en serie con otros dos contactos en paralelo y todo esto en paralelo con dos contactos en serie.
En la programación de este circuito, el primer paso que daremos es iniciar la prime ra secuencia con los dos primeros contactos en paralelo:

LI0.0
OI0.1

Hasta ahí tenemos nuestro primer bloque al cual llamaremos **"A".**
Luego iniciamos una nueva secuencia con los dos siguientes contactos en paralelo:

LI0.3
OI0.4

Con esto formamos nuestro segundo bloque, al cual llamaremos "**B**". Si nuevamente realizamos un análisis de las operaciones **STACK** de la **CPU** tendremos lo siguiente:

a)Se escribe el estado informativo del contacto **I0.0** de la primera línea , quedando almacenado en el **STACK Nº1**.

b)Con la segunda linea **I0.1**, la **CPU** hace una conexión en paralelo **(OR)** y el resultado se sobreescribe en el **STACK Nº1**. A este resultado lo llamaremos "**A**".

c)Con el siguiente inicio de secuencia (**L**), la **CPU** automáticamente tiene que desplazar la información "**A**" contenida en el **STACK Nº1** hacia el **STACK Nº2**. De esta forma queda libre de contenido el **STACK Nº1** y sobre el se escribe la información correspondiente a la línea 3, que es el inicio dela siguiente secuencia.
La **CPU** lee la información correspondiente al estado del contacto **I0.3**, y la escribe en el **STACK Nº1** ya vacío.

d)La CPU lee el estado del contacto **I0.4**, correspondiente a la cuarta línea, y hace una conexión en paralelo (**OR**) con el estado de la tercera línea, quedando sobreescrito el resultado en el **STACK Nº1**. A este resultado lo llamaremos "**B**". De esta forma queda actualizada la información sobre el **STACK Nº1**
A continuación haremos directamente la conexión en serie del bloque " **A**"y el bloque "**B**". Por lo tanto, a continuación escribimos directamente la operación A, tal como se muestra en la siguiente secuencia:

LI0.0
OI0.1
LIO.3
OI0.4
A

Con esto automáticamente el bloque **"A"** queda fundido con el bloque **"B"** en conexión serie. A este resultado le llamaremos **"C"** y quedará almacenado en el **STACK Nº1**, siendo vaciado el **STACK Nº2.**
Posteriormente abrimos una nueva secuencia con la operación **LOAD (L).** Con este nuevo **L** desplazamos el contenido del **STACK Nº1** hacia el **STACK Nº2**, quedando registrado el contenido del contacto I0.2 en el **STACK Nº1**. Luego se hace una conexión en serie con el contacto I0.5, quedando sobreescrito este resultado, al cual llamaremos **"D"** en el **STACK Nº1**. Luego realizamos una conexión en paralelo con el contenido del **STACK Nº2** y el contenido del **STACK Nº1**, operación que mostramos a continuación:

LI0.0
OI0.1
LI0.3
OI0.4
A
LI0.2
AI0.5
O

Con esto quedarán fundidos los contenidos del **STACK Nº2** (C) y **STACK Nº1** (D) en una nueva resultante que llamaremos "E", la cual quedará almacenada en el **STACK Nº1.**

Aquí dejamos bastante claro que podemos abrir muchísimos más de 8 paréntesis. Lo que podemos tener como máximo son 8 paréntesis abiertos. Como hemos abierto 3 paréntesis, solamente hemos ocupado 2 STACK debido a que hemos abierto y cerrado, por lo tanto, mientras más rápido se cierren los paréntesis, mayor será la cantidad de **STACK** que podremos ocupar.

Se pueden tener máximo 8 abiertos, aunque en realidad 7, porque uno es el básico que siempre se necesita.

Por lo tanto, es más fácil, para luego seguir el esquema, si lo más rápido se cierran los paréntesis.

También es muy importante saber, que cuando se hace esta operación, siempre se hace del **STACK Nº1** al **STACK Nº2.**

El **PS4 100** tiene capacidad para 1000 líneas de instrucción. En este programa hemos ocupado 8, por lo tanto, existe una amplia posibilidad para realizar programas.

Si hubiésemos programado a través de merquer, tal como lo vimos en la primera opción de programación, en lugar de 9 líneas de instrucción, hubiésemos ocupado 13, por lo tanto, más lento resulta el programa.

MODULOS INTERNOS DEL PS4 - 100.

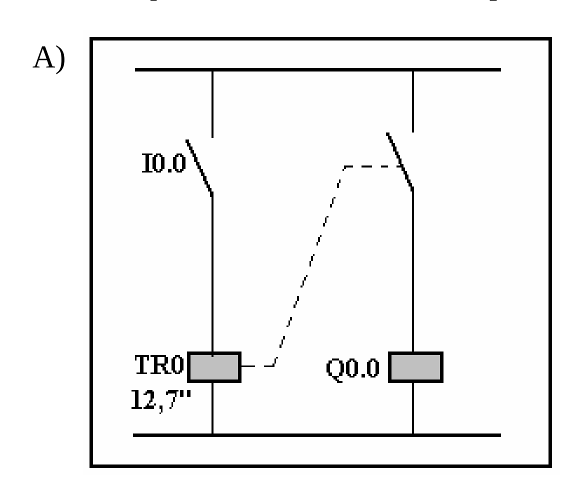
En una maniobra eléctrica existen algunos elementos que siempre son muy repetitivos y en el **PS4 - 100** se han dispuesto 4 funciones repetitivas que son las más usuales, puesto que siempre se utilizan. Estas funciones se han dispuesto para no tenerlas que inventar y son las siguientes:

- **32 Temporizadores. (TR)**
- **32 Contadores ©.**
- **32 Comparadores. (CP)**
- **32 Registros de desplazamiento. (SR)**

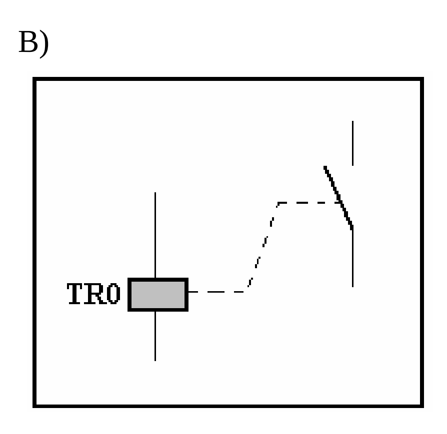
1.-TEMPORIZADORES (TR).

Si bien el temporizador no realiza una operación lógica, la incluir emos dentro de este grupo por ser de gran uso en los diagramas.
El PS4 - 100 incluye la posibilidad de programar hasta 32 temporizadores.
En un esquema convencional el temporizador se representa de la siguiente manera:

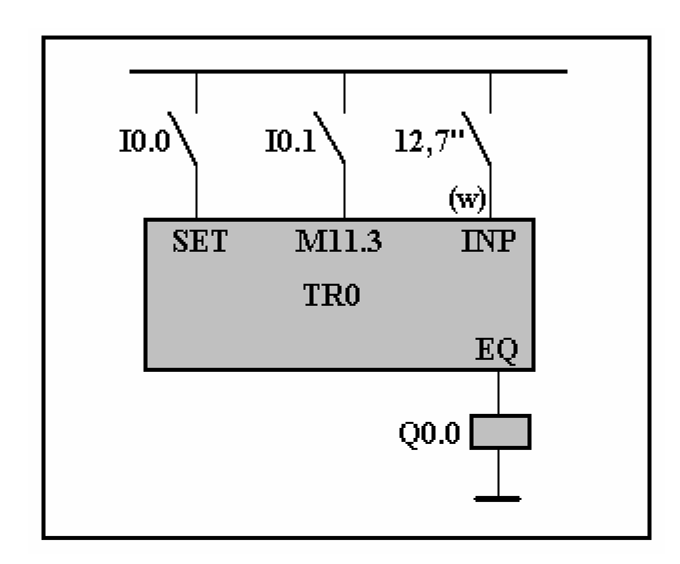
A)



B)



En la figura A se muestra el circuito del temporizador, el cual, podemos ver, consiste en una bobina que comanda un contacto temporizado. A esta bobina tenemos que señalarle bajo que condiciones debe actuar (arrancar), cuanto tiempo debe tardar y sobre que salida debe actuar una vez transcurrido el tiempo programado.
En la figura B tenemos al contacto **N.A**. I0.0, el cual actúa sobre la bobina del temporizador y el contacto temporizado **N.A.** actúa sobre la salida Q0.0.
Si nosotros necesitamos que este temporizador quede ajustado a 12,7 segundos, este tiempo se calibra a través del potenciómetro frontal.
La forma real de representar un temporizador, en un esquema de contactos, se muestra en la figura siguiente.



Podemos apreciar que nuestro temporizador posee tres entradas y una salida.
La primera entrada se denomina **SET** (arranque), y a través de ella se le da partida al temporizador.
La segunda entrada es **INPUT** (I), la cual tiene que ser de longitud palabra (W) y a través de ella defino el tiempo.
La salida es **EQUAL** (EQ) y en ella se conecta el elemento sobre el cual debe actuar el temporizador.
Existe en el **PS4 - 100** una tercera entrada denominada **STOP** y que luego explicaremos para que sirve.
Si queremos trabajar con el primer temporizador, se escribe la instrucción **TR0** y luego se pulsa la tecla **ENTER**. En la pantalla se abrirá una listado vertical que mostrará lo siguiente:

SET
STOP
I(W)
EQUAL

Es decir, en la pantalla aparecen los 4 parámetros constantes del temporizador ya listos, lo que significa, que nosotros solo debemos programarlos.

SET : I0.0
STOP :
I(W) : KW127
EQUAL : Q0.0

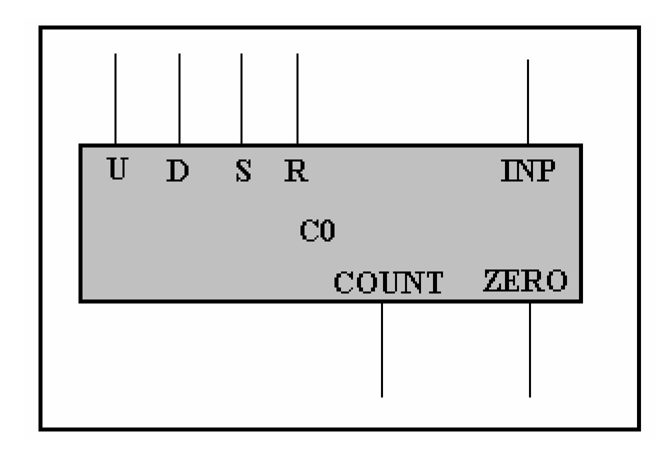
Por ejemplo, el **SET** es el I0.0. El **STOP** no lo utilizamos y por lo tanto, no escribimos nada.
En el **INPUT WORD** tenemos que escribir 12,7. Como los 12,7 corresponden a un valor fijo, y los valores fijos de denominan constantes **(K)**, al escribir los 12,7 debemos anteponer la letra **K** para indicar que este valor es constante. Además , como me exige que tiene que ser una constante de longitud palabra, tengo que indicar que es una constante Word de 12,7, es decir, **KW127**. Es importante destacar que el tiempo no se escribió en 12,7, sino que en 127, puesto que los temporizadores del **PS4 - 100** tienen una base de tiempo de 0.1 segundos, por lo tanto, todo valor expresado en segundos se debe multiplicar por 10.
El **EQUAL** (EQ) se escribe Q0.0.
Así de fácil es como se programa un temporizador.

FUNCIONAMIENTO DEL TEMPORIZADOR

Si nosotros, cerramos el contacto I0.0 y le otorgamos señal a la entrada **SET**, el temporizador inicia su cuenta: 1, 2, 3, 4, 5, etc. Si luego es abierto el contacto I0.0, el contador automáticamente se vuelve a cero.
Si vuelvo a cerrar el pulsador I0.0, el temporizador vuelve a iniciar la cuenta desde cero, es decir, 1, 2, 3, 4, 5, etc: Si mantenemos el contacto I0.0 cerrado, y luego a través del contacto I0.1 le damos un impulso a **STOP**, el temporizador se detiene justo ahí donde estaba en la cuenta. Si luego quitamos el impulso de **STOP**, se reinicia la cuenta hasta llegar a 12,7 segundos. Por lo tanto el **STOP** sirve para hacer una detención temporal de la cuenta, sin volver a cero, es decir, para hacer una pausa.
En cambio, el **SET** sirve para arrancar y para volver a cero. Si algún día, en el temporizador necesitamos esta función, la podemos utilizar, de lo contrario dejamos la línea en blanco.
En el caso de querer ocupar esta función, si **STOP** depende del merquer M11.3, si esta es la condición de hacer la pausa, se escribe solamente M11.3. Si **STOP** es una entrada I.01, pues se escribe I0.1. Cualquier contacto de un bit es lo que se necesita.
En el **PS4 - 100,** el máximo valor de tiempo que podemos programar en la entrada I(W) del temporizador es de 65.535 y como la base de tiempo de cada temporizador es de 0,1 seg., esto equivale a 6553,5 segundos (65.535 x 0.1), lo que equivale a 1,82 horas.

CONTADORES

Esta instrucción, también se incluye dentro de este grupo, por ser muy utilizada en los diagramas de programación. El **PS4 - 100** incluye la posibilidad de programar hasta 32 contadores.
Los contadores se designan con la letra C, es decir, tenemos desde el **C0** hasta el **C31**.
El esquema de un contador se muestra en la figura siguiente.



Las entradas son:

U (UP).
D (DOWN).
S (SET).
R (RESET).
INP (INPUT).

Las salidas son:

COUNT.
ZERO.

Cuando se conecta el **PLC** el contador está en cero. Si el contador está en cero, esta activada la salida cero.
Si luego activamos la entrada **UP**, el contador ira incrementando su valor de la siguiente forma:

Primer impulso en UP = 1
Segundo impulso en UP = 2
Tercer impulso en UP = 3
.
.
.
.
Ultimo impulso = 65.535

Por otro lado, por cada impulso aplicado a la entrada **DOWN** el contador va disminuyendo la cuenta. Si llega a cero, se vuelve a activar la salida zero, puesto que esta solamente se activa cuando el contador está en cero.

Cuando el contador está en cualquier otro estado, la salida zero no está activada.

Acá nosotros tenemos la posibilidad de asignarle un valor directo al contador. Por ejemplo, queremos que el contador de golpe asuma el valor 1000, es decir, que inicie la cuenta a partir de esta cifra. Para lograr esto, escribimos en la entrada **INPUT** el valor 1000 y luego activamos la entrada zo **SET**. Esto permite que el contador inmediatamente asuma el valor 1000. Una vez lista esta operación, comienza aplicar impulsos a la entrada **UP**, con lo cual la cifra del contador comienza a incrementarse:

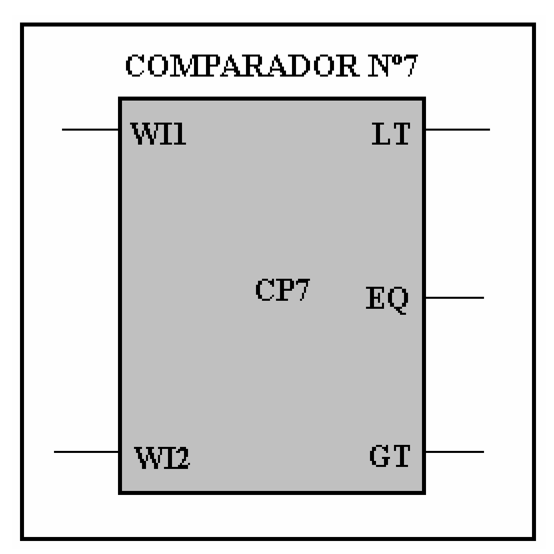
Primer impulso = 1001
Segundo impulso = 1002
Tercer impulso = 1003
.
.
.
.
etc.

Luego, si aplicamos impulsos en la entrada **DOWN**, el contador comienza a descender su cuenta, por lo tanto, estamos en presencia de un contador bidireccional.
Ahora, si está el contador en 1000 y aplico 1000 impulsos en la entrada **DOWN**, el contador llega a cero y se activa nuevamente la salida **ZERO**.
Si a continuación., en cualquier momento de la cuenta, estando el contador en la cifra que esté, aplico un impulso en la entrada **RESET**, el contador vuelve a cero y se activa la salida **ZERO**, por lo tanto, por la salida **COUNT** obtenemos la salida real que en todo momento está entregando el contador, lo que nos permite utilizarlo para que cuente de cero hacia arriba.
Al contador no le podemos fijar un valor de cuenta, por ejemplo que cuente hasta 1000 y se detenga. Para lograr esto, tendríamos que utilizar otro bloque llamado co mparador, entonces le podríamos pedir que nos compare el valor 1000 con el valor 50, pero este tipo de bloque lo veremos más adelante.
Si tenemos un solo valor que contar, tenemos la posibilidad de no utilizarlo como contador de subida o ascendente, sino que como contador de bajada o descendente.
Para este último caso, si en la entrada **INP (W)** escribimos el valor 1000 y luego aplico un impulso a la entrada **SET**, automáticamente la salida **COUNT** se pone en 1000. Posteriormente, en lugar de aplicar impulsos a la entrada **UP** los aplico en la entrada **DOWN**, la resultante es bastante simple, pues cuando se active la salida **ZERO** es que hemos llegado a nuestro valor. Por lo tanto, si tenemos un solo valor, es mejor utilizarlo como contador descendente, de tal forma que cuando se nos activa la salida **ZERO** es que tenemos nuestra función cumplida.
El contador que hemos analizado es de longitud palabra y por lo tanto, cuenta de **0** a **65535.**

COMPARADORES (CP).-

En el contador hemos visto que la entrada **INPUT** es de longitud palabra, por lo tanto puede contar hasta 65.536 (2^{15}), es decir, de 0 a 65.535. Si luego, con mucha paciencia comenzamos a incrementar la entrada **UP** hasta llegar al valor 65.535 y luego pulsamos nuevamente la entrada **UP**, vamos a notar que la salida del contador vuelve a cero y se reinicia el ciclo, es decir, el contador no se queda detenido. Esto significa que es como el cuenta kilómetros de los vehículos. Por lo tanto, los contadores pueden contar hasta 65.535.

Los comparadores se designan como **CP** y en total son 32 (0 al 31) y su función es la de comparar dos valores aplicados a sus entradas, cada uno de longitud palabra, es decir, 65.535.



WI1 : Entrada 1 (longitud word)
WI2 : Entrada 2 (lon gitud word)
GT (Greater than) : Mayor que (I1 > I2)
EQ (Equal) : Igual a (I1 = I2)
LT (Less than) : Menor que (I1 < I2)

El modulo compara los valores en las entradas de palabra (Word) I1 e I2 y a continuación pone las salidas conf orme a la tabla de trabajo.

Para entender su funcionamiento, supongamos que en la entrada INPUT 1 **(WI1)** ponemos el valor 1 y en la entrada INPUT 2 **(WI2)** ponemos el valor 2.
Si observamos el comparador podremos notar que posee tres salidas: **LT** (1<2), **EQ** (1=2) y **GT** (1>2) **.**
Si el valor de la entrada **INPUT 1** es menor que el valor de la entrada **INPUT 2**, se activa la salida **1<2**.

Si el valor de la entrada **INPUT 1** es igual al valor de la entrada **INPUT 2,** se activa la salida **1=2**.

Si el valor de la entrada **INPUT 1** es mayor que el valor de la entrada **INPUT 2,** se activa la salida **1>2**.

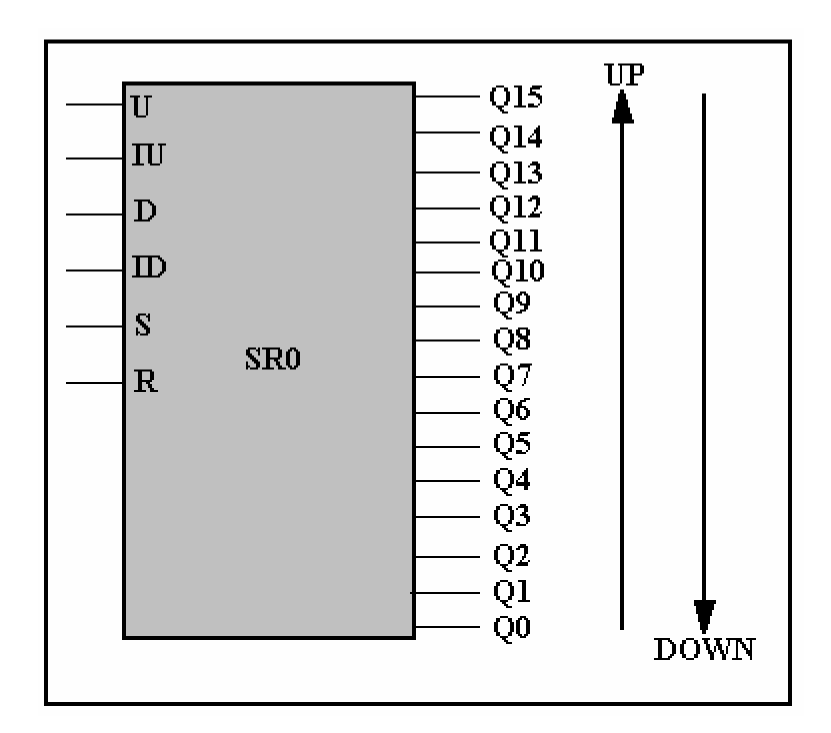
También podríamos ingresar a la entrada **INPUT 1** del comparador, la salida de un contador y a la entrada **INPUT 2** una constante de longitud palabra. Si el valor de salida del contador es menor que el valor de la constante de longitud palabra, se activa la salida **1< 2.**

Cuando el valor de salida del contador llega exactamente al valor de la constante de longitud palabra, se activa la salida **1=2**.

Cuando la salida de contador tiene un valor mayor que el de la constante de longitud palabra, se activa la salida **1>2**. Por lo tanto, a través de un comparador puedo comparar valores hasta 65.535.

Es importante destacar que también se podría realizar la comparación de las salidas de dos contadores.

REGISTROS DE DESPLAZAMIENTO (SR).



En el tenemos 6 entradas y 16 salidas (0 al 15).

Las entradas son:

UP
INFOUP
DOWN
INFODOWN
SET
RESET

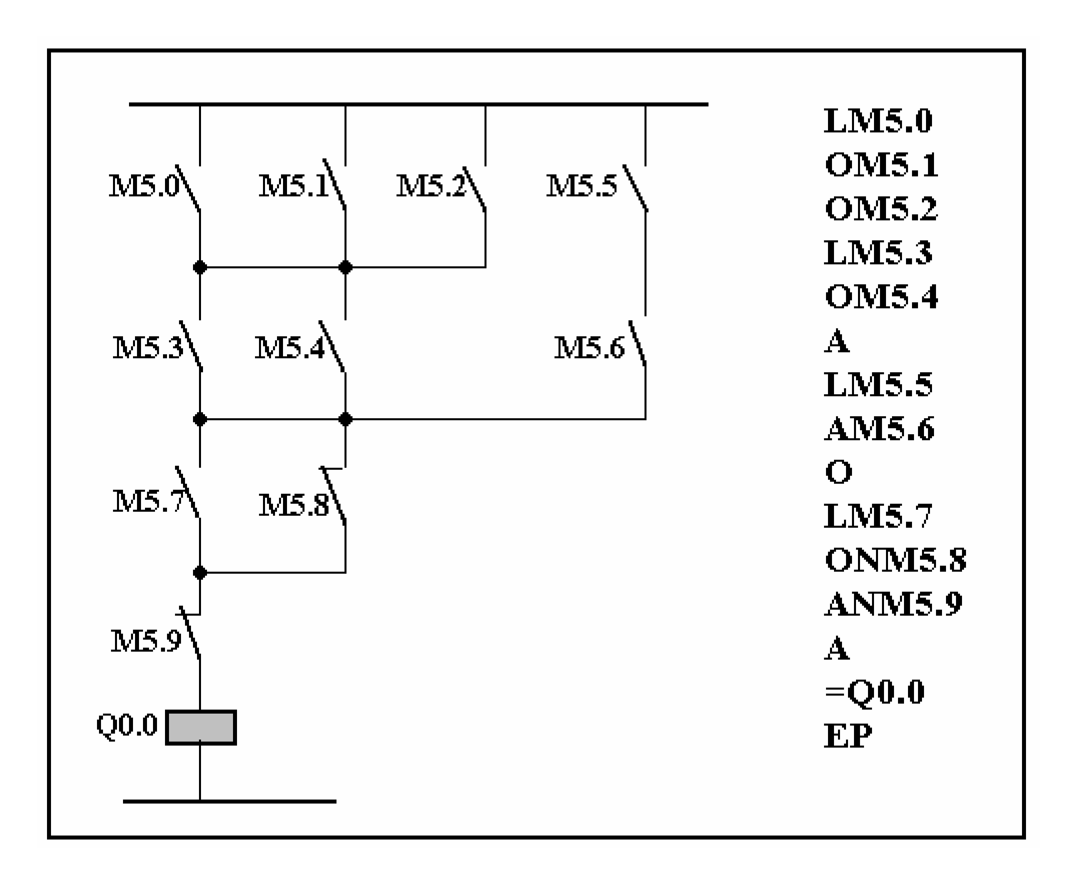
Las salidas son:

Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
Q9
Q10
Q11
Q12
Q13
Q14
Q15

Su forma de trabajo es la siguiente:
Se aplica información en la entrada **INFOUP (IU).**
Cuando se inicia el trabajo con el SR, obviamente todas las salidas están en cero. En estas condiciones, si aplicamos un pulso en la entrada **UP**, es la información que está en la entrada **IU** (0 o 1) la que se desplaza. Si por ejemplo, en la entrada **IU** tenemos un bit 1 y pulsamos la entrada **UP**, este 1 puesto en la entrada **IU** se desplazará hacia la salida Q0, manteniéndose las restantes salidas en 0.
Si mantenemos este 1 en la entrada **IU** y damos nuevamente un impulso en la entrada **UP**, entonces el 1 que teníamos en la salida **Q0** será desplazado hacia la salida **Q1** y el que teníamos en la entrada **IU** se desplazará hacia la salida **Q0.**
Ahora, si en la entrada **IU** tenemos un bit 0 y damos nuevamente un impulso en la entrada **UP**, el 1 que teníamos en la salida **Q1** se desplaza hacia la salida **Q2** y el que estaba en **Q0** se desplaza hacia **Q1**. De esta forma, el 0 que teníamos en la entrada **IU** se desplazará hacia la salida **Q0.**
Como podemos ver la forma de trabajo es bastante simple.
Supongamos que ahora se aplica información en la entrada **INFODOWN (ID)** y que nuevamente todas las salidas están en cero. Por cada pulso que apliquemos en la entrada **DOWN**, la información presente en la entrada **ID** es la que se desplaza

hacia **Q15** y por cada pulsación aplicada sobre la entrada **DOWN**, esta información se ira desplazando hacia abajo.
Estos registros de desplazamiento se llaman **SR (Shift Register)** y en el **PLC 100** tenemos 32, es decir, desde el **SR0** hasta el **SR31**.
Cada uno de estos **SR** posee 16 pasos, es decir, 16 salidas. Si en algún momento llegamos a requerir más de 16 pasos, podemos enganchar el último paso al **SR** siguiente, lográndose de esta manera obtener registros de 16 x 32 pasos

PROGRAMACION 2



Regla: Siempre verifique que la cantidad de operaciones **Stack** sea siempre igual a la cantidad de operaciones Load menos uno. En el caso del circuito programado, se puede verificar que existen 4 operaciones **Load** y 3 operaciones **Stack**, por lo tanto la programación está bien realizada.

GUIA OPERACIONAL DEL PROGRAMA

El **PS4 100** posee 2 puertas de comunicación. Las dos tienen el **INTERFACE RS485** y tienen el punto de **Quorum** superior **K1**, que es el protocolo para poner los esclavos.
Debido a que en esta puerta tenemos un protocolo fijo y no un protocolo transparente y tenemos, además, el **INTERFACE RS485.**

El problema es que un PC no tiene entrada o salida **RS485** y mucho menos tiene **SUCCO**, por lo que se necesita un cable que por un lado convierta el **RS485** en **RS232** y por otro lado convierta la señal **SUCCO** en protocolo transparente. Por lo tanto, el **PS4 100** siempre requerirá este cable con convertidor, el cual por supuesto, viene incluido en la máquina.

Este cable conector (**RS485)** se conecta a la primera puerta de comunicación y a partir de la segunda puerta de comunicación se conectan los esclavos.

La salida de este cable conector **(RS232)**se conecta en la salida serie del PC. El **MOUSE** siempre ocupa el **COM1,** por lo tanto este cable conector deberá ser conectado en el **COM2.** Si el PC no tiene **COM2** nos veremos obligados a trabajar sin MOUSE.

En nuestro caso no tenemos, **COM2** por lo tanto, el cable conector deberá ser conectado en el **COM1.**

El programa se llama **SUCCOS SOFT** y el archivo para entrar se llama **SUCCOS 3**. Este archivo a quedado guardado en un subdirectorio que se llama **SUCCO**. En este subdirectorio estarán todos los archivos necesarios para trabajar el sistema. Desde este directorio base, se escribe **SUCOS3** y entramos en el **SUCCO SOFT**. Luego seguimos con ENTER. En este punto se rápidamente en pantalla que estaba elegido el **COM 1**. Si nosotros, algún día, queremos cambiar a otra puerta, existe la posibilidad de ir a elección de sistema a través de **F7**. Esto hay que hacerlo una sola vez en la vida, puesto que aquí se define el entorno del **Softward** .

A todos los menú y submenú se accede a través de las teclas de función, por lo tanto, al accionar la tecla de función **F7,** aparece en pantalla **"Elección del sistema".** Si desde aquí, presionamos la tecla de función **F4**, en la pantalla aparecerá el submenú **"Elección de la interface".** Desde aquí podemos elegir el **com1,** el **com2** o la **EPC334**.

Esta última es una tarjeta que antes incluíamos en el **PLC** y que tenía directamente una salida **RS485**. El disponer de esta tarjeta nos permitía trabajar sin convertidor. Lo que pasa es que la tarjeta **EPC334** es mucho más cara que el cable de interface y por ello a sido eliminada.

Nosotros hemos seleccionado el, com1 para lo cual fue accionada la tecla de función **F2**, de esta manera queda determinado que siempre vamos a actuar con el com1. Si el día de mañana queremos trabajar con el **com2,** pulsamos la tecla de función **F3**. Por el momento el formato lo dejaremos en el **com1**.

En este mismo submenú **"elección del sistema "** tenemos la posibilidad, a través de la tecla de función **F3**, de elegir la impresora. Aquí uno elige la que más se asemeja a la que tiene para trabajar. Si hacemos la elección y me da error, es porque el sistema intento comunicarse una impresora similar y no la encontró.

En **"Elección del sistema"** también podemos elegir el idioma a través de la tecla de función **F5**.

F1 es la tecla de función del menú principal, el cual es el menú desde donde hemos partido. Este menú nos permite tener acceso a los diferentes submenús. Uno de estos submenús era , como ya hemos visto, **"Elección del sistema".**

El más importante de los submenú, es el de programación **(F1),** puesto que desde aquí se puede programar, tanto en **LDI** como en **EDC** .

Desde todos los submenú, menos del menú principal, la tecla de función **F1** es siempre para ir un paso hacia arriba, es decir, hacia el menú superior y todas las otras teclas son para ir hacia abajo.
Como ayuda diremos lo siguiente, si el menú que aparece esta escrito con mayúsculas, es que existe un submenú, pero si esta escrito con minúsculas, es el último submenú.
Con F1 volvemos al **" Submenú de programación ",** por lo tanto , desde aquí, a través de las teclas de funciones **F2, F3** y **F4** podemos seleccionar tres alternativas de lenguaje para la programación. Una de estas alternativas es la de trabajar en **" Lista de instrucciones ",** que es lo mismo que hemos hecho antes. También podemos programar directamente en **"Esquema de contactos",** que no es otra cosa que una lista de instrucciones.
Los tres sistemas de programación son completamente compatibles. Podemos trabajar en lista de instrucciones, luego ver el programa en esquema de contactos e imprimirlo en esquema de bloques funcionales o viceversa.
Desde el submenú de programación, presionamos la tecla **F2** (programación en LDI). A continuación le asignaremos un nombre al programa, aunque el me va a preguntar por dos nombres para dos archivos diferentes. Uno es el programa fuente y el otro, la lista de referencias. El nombre que le asignaremos a estos dos archivos lo podemos elegir libremente, pero debe contener un máximo de ocho dígitos.
Supongamos que al programa fuente lo llamamos **"TEST"** y a la lista de referencias también la llamaremos **"TEST".** Esto no es problema, porque el sistema añade automáticamente la extensión **Q3** al nombre del programa fuente y la extensión **Z3** al nombre de la lista de referencias, por lo tanto, se han creado dos archivos que se llaman **"TEST Q3"** y **"TEST Z3"** . Es preciso indicar que para compilar el sistema requiere de estos dos ficheros.
Para editar el programa debemos accionar la tecla **F2**, apareciendo la pantalla completamente vacía. Luego se activa la tecla de función **F2** (abrir bloque), puesto que tenemos que abrir como mínimo un bloque.
Si establecemos una equivalencia con un libro, podemos decir que un bloque es el equivalente a un capitulo de este libro y escribir un libro capitulándolo es importante, pues si tenemos un libro escrito en forma totalmente continuada, desde la pagina 0 hasta la pagina 5579 y debemos buscar algún término en él, tendríamos que revisar desde la pagina 0 hasta la pagina 5579.
Para evitar esto, lo mejor es escribirlo por capítulos, lo cual , evidentemente haría más fácil el buscar un determinado término. Mientras más capítulos tengo, más fácil resulta encontrar ese término.
Un bloque no es más que el equivalente a un capitulo de un libro, y no tiene ninguna importancia en la longitud del programa en el PLC, puesto que solamente nos ocupará espacio en el disco duro del PC.
Como se puede apreciar, cada bloque tiene una cifra de cinco dígitos, por lo tanto, teóricamente podemos abrir 99.999 bloques. A cada bloque le podemos asignar un nombre, de tal forma que nos 0000 resulte más fácil ubicarlo. Por lo tanto, inicio el bloque con cualquier nombre y luego enter. A continuación empieza la línea y ésta se inicia con comillas. Todas las líneas que se inician con comillas son líneas de

comentarios y en ellas se puede escribir cualquier comentario acorde a lo que se va a ejecutar. Si con una línea no tenemos espacio suficiente, en la línea siguiente se abre nuevamente comillas y se continua escribiendo comentarios. también podemos dejar líneas en blanco. Todo esto no ocupa ningún espacio en el **PLC**, solamente ocupa espacio en el disco duro del PC.
Si a continuación queremos empezar a programar, es cuando realmente debemos hacer uso de instrucciones.
Veamos uno de los programas desarrollados en la clase:
En la línea **001** anotamos la instrucción **LI0.0** Se puede apreciar que se ha escrito todo junto y que el mismo hace sus espacios para que quede más vistoso. El cursor salta a la mitad de la pantalla. En esta posición podemos escribir un comentario relacionado con éste operando (por ejemplo, “ pulsador de marcha “), claro está que este comentario es optativo y que su longitud puede ser como máximo, la mitad sobrante de la pantalla.
En la línea siguiente **(002**) desarrollamos la instrucción **OI0.1** y si no queremos escribir comentarios, pasamos a la línea siguiente **(003)** con la instrucción **=M5.1.**
Ahora, si queremos dejar esto más llamativo, dejamos la línea **004** en blanco, con lo cual logramos que los bloques queden más claramente identificados.
En la línea **005** escribimos la instrucción **LI0.2** y en la línea **006** la instrucción **OI0.3.** El sistema va probando si las distancias entre líneas de instrucciones son correctas. Lo que el sistema no puede controlar es un error de conexionado. Imaginemos que conectamos el contacto I0.2 en paralelo con el potenciómetro. El **I0.0** es una entrada de **BIT** y el potenciómetro es una entrada analógica **de BYTE** y ocurre que conectar un **BIT** con un **BYTE** en paralelo no es posible. Ahora el sistema no detecta el error en el momento dela escritura, puesto que el si ntaxis es el correcto, si no que lo detecta en el momento de la compilación, puesto que es en ese momento cuando se entera que hemos tratado de mezclar un **BIT** con un **BYTE**.
En el momento de escribir las instrucciones, línea por línea sintaxicamente sabe si se ha escrito bien o no, por lo tanto, de existir un error, éste deberá ser corregido para poder seguir adelante.
Suponiendo que no hay errores de sintaxis, en la línea **007** escribimos **=M5.2.** Nos saltamos la línea **008**. En la **009** escribimos **LM5.1** y en la **010** escribimos **AM5.2**, para luego, en la línea **011** escribir **=Q0.0.**
En un **PLC** los merker o relés internos no los podemos ver. Lo que sí podemos visualizar son las salidas, dado que estas tienen led´s, entonces la salida, en lugar de asignarla a un merker, como estaba en el ejemplo, le asignaremos un **Q**.
Todo final de programa deberá llevar la instrucción **EP**.
Una vez finalizado el programa, se requiere salvarlo y salvar está en el menú superior, por lo tanto para ir a éste menú, debemos pulsar la tecla de func ión **F1**.
Una vez ubicados en el menú superior debemos presionar la tecla de función F4, con la cual podremos salvar los datos. Ahora, él nos deja la opción de salvar con un nombre actual o con un nombre diferente. Como lo queremos salvar con el nombre actual, presionamos la tecla de función **F2.**
Para entender la finalidad de querer salvar con un nombre diferente, imaginemos que queremos hacer una variante de lo ya programado, pero sin perder el original.

Entonces editamos lo programado a través de la tecla de función F2, luego modifico las líneas que me hacen falta, vuelvo otra vez a **F1** para retornar y ahora pido salvar datos, pero ahora no los salvo con el nombre actual, sino que salvo el programa fuente con **TEST 1** o cualquier otro nombre, la lista de referencias también con otro nombre. De esta forma tengo dos programas salvados; el original, que queda tal cual estaba antes y el modificado, que queda con un nuevo nombre.
Este es el motivo por el cual podemos salvar la información programada con el nombre actual o con un nuevo nombre.
En nuestro caso el programa lo hemos salvado con el nombre actual.
Posteriormente , a través de la tecla de función **F1** retornamos al menú principal, desde donde realizaremos la compilación. Lo que ahora hemos escrito son dos ficheros. Un fichero es el programa fuente, el cual, como hemos visto, contiene un monton de líneas de comentario, líneas en blanco y líneas de operando. Para el **PLC**, todo esto, obviamente no tiene ningún sentido y por lo tanto no sabe que hacer con ello. El **PLC** solo entiende los programas compilados, es decir, programas desarrollados en lenguaje de máquina. Por lo tanto, existe la necesidad de compilar este programa y esto lo logramos a través de la tecla de función **F6**. Cuando es activada la tecla de función **F6**, se nos pregunta cual es el programa fuente que queremos compilar, es decir, traducir a lenguaje de máquina, y el ya me da el nombre, por lo tanto, solamente lo debemos confirmar a través de enter y luego confirmar que se encuentra en C. Después de esto me indica si se han encontrado fallas.
Hemos utilizado 10 instrucciones, para lo cual se han ocupado **36 BYTE** en el **PLC**, quedando aún disponibles **3644 BYTE** para seguir programando.
Una vez compilado el programa, es decir, traducido a lenguaje de máquina, éste debe ser traspasado al **PLC**.
Para lograr esto, debemos ir al menú principal, es decir, volver al punto de partida. Desde éste menú, activamos la tecla de función **F4** para solicitar la transferencia y desde el submenú de transferencia presionamos la tecla de función **F2**, con lo cual se concreta la transferencia desde la unidad al **PLC**. Es importante destacar que solo se pueden transferir al **PLC** programas ejecutables (compilados).
Durante la compilación, del programa fuente, llamado **TEST Q3** y de la lista de referencias, llamada **TEST Z3**, el ha hecho un programa llamado **TEST P3**, donde **P** es el programa compilado y solamente este programa compilado es el que podemos transferir. El programa compilado recibe automáticamente el mismo nombre del programa fuente, por lo tanto , con **ENTER** lo confirmamos y luego le indicamos que se encuentra en el disco **C.**
Si el hubiese detectado algún error, como por ejemplo, que no estuviese conectado el cable de interface **RS485** que une a la unidad con el **PLC**.
A continuación salimos del menú de transferencia a través de la tecla de función **F1** y nos vamos al submenú de **TEST DE PUESTA EN MARCHA**. En esta pantalla el nos indica cual es el aparato base y si tuviéramos conectados nuestros tres esclavos, nos indicaría que tipo de esclavo tengo conectado. También nos indica que el aparato base es un **PS4-111**, que tiene tensión (Power) y que se encuentra en **RUN**, es decir, ejecutando el programa.

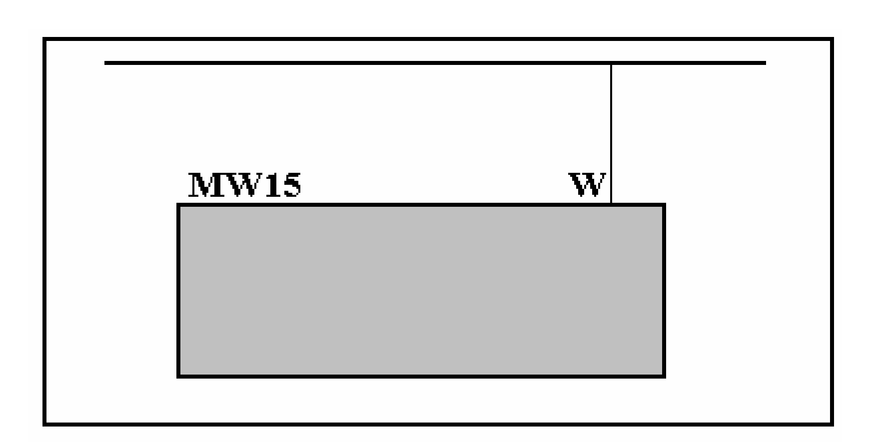
Desde aquí podemos definir tres formas de arranque del **PLC**:
Una es el arranque automático normal, lo que significa que si en el momento de aplicar tensión, el PLC se encuentra cargado con un programa en su interior, arranca automáticamente y lo hace con todos los valores que previamente habíamos almacenado.
Una segunda forma de arranque es el arranque automático. Esta forma de arranque es importante, puesto que, es factible , que en algún momento sea necesario iniciar el arranque completamente desde cero, es decir, que nos borre absolutamente toda la información, con todos los merker en cero.
La tercera posibilidad es el arranque **HALT** , vale decir, que cuando nosotros apliquemos tensión necesitemos que el **PLC** no arranque, que se mantenga detenido.
Por ejemplo, para el caso de una prensa, puesto que aquí es obligatorio que desde el **PLC** alguien de la orden para el arranque.
De acuerdo a esto, existen tres diferentes formas de arrancar. La normal es siempre la primera.
Desde aquí, lo que también podemos hacer es detener o arrancar el **PLC.** Hemos dicho que, debido a la primera condición de arranque, el **PLC** ha arrancado, pero si lo queremos detener, lo podemos hacer a través de la tecla de función **F3**. El arranque lo conseguimos a través de la tecla de función **F2.**

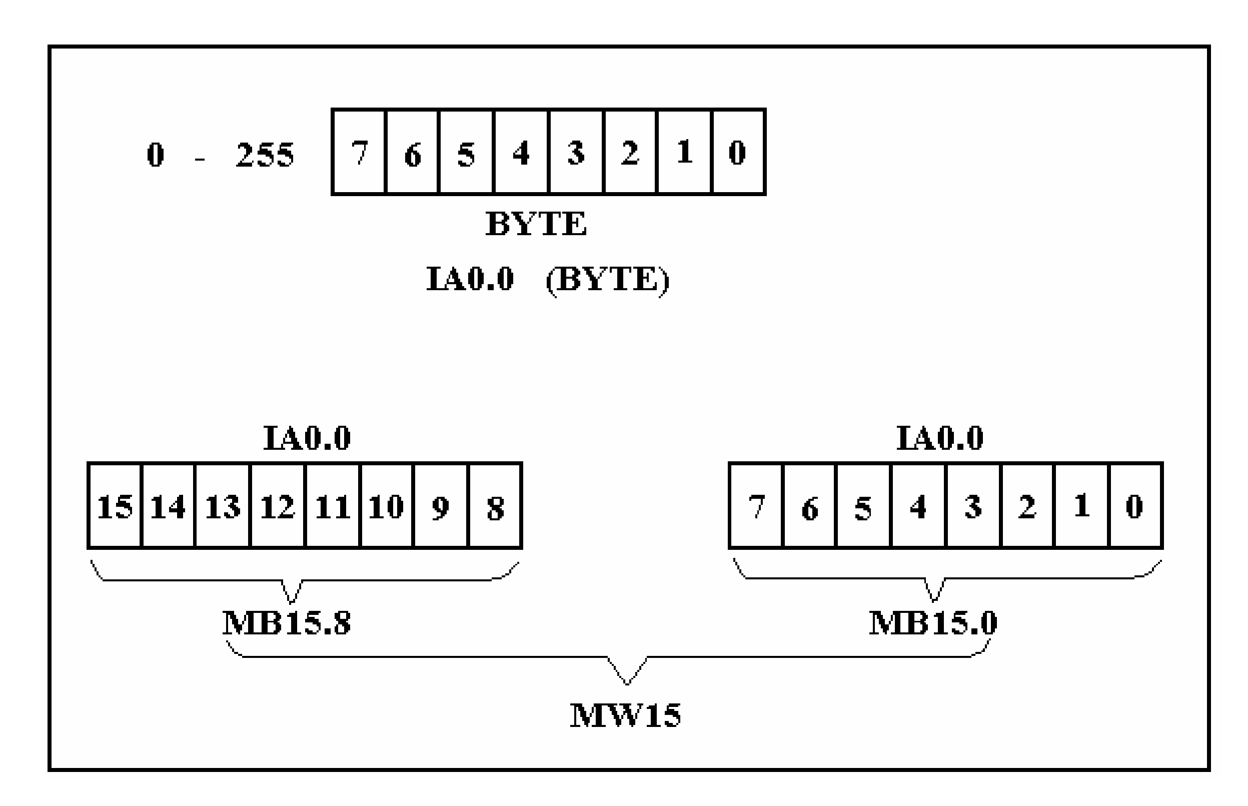
ENTRADA ANALOGICA

Analizaremos a continuación el potenciómetro del **PS4**, el cual utilizaremos como una entrada interna analógica.
Hemos dicho que la entrada analógica del **PS4 - 111** tiene 8 bit de resolución, por lo tanto ocupa exactamente 8 bit. Si cada uno de estos bit está en cero, el valor decimal que de aquí resulte también será cero. En cambio, si cada uno de estos ocho bits está en uno, el valor decimal que obtendremos será **255 (2^8)**, por lo tanto, si llevamos el potenciómetro a un extremo, el valor obtenido será cero, mientras que en el otro extremo obtendremos el valor **255**. En cualquier posición intermedia obtendremos un valor intermedio resultante de la combinación de ceros y unos.
Recordemos lo siguiente, para asignar un valor fijo, lo hacíamos a través de la constante **KW**, por ejemplo**, KW12,7** o **KW50.**

CONVERSION DE UNA ENTRADA ANALOGICA DE LONGITUD BYTE EN LONGITUD WORD.-

Para lograr esto, recordemos como funcionen los registros internos (merker) que vimos antes.
Los merker son registros internos de longitud palabra.
Nosotros podemos ocupar cualquiera de los 32 merker disponibles, siempre y cuando no estén siendo ocupados en otro sitio. Pongamos como ejemplo el caso del merker word 15 **(MW15):**





Si nos piden escribir el valor 138 y luego comparar esta cantidad con 00000138, podemos decir que la cantidad es la misma, es decir, no cambia. Si el valor de nuestra entrada analógica IA0.0 lo escribimos en el **MB15.0** y nos aseguramos que los otros ocho bits del **MB15.8** estén en cero, seguiremos manteniendo exactamente el valor de nuestra entrada analógica.

Por lo tanto, si nuestra entrada analógica deja todos los bits del **MB15.0** en ceros, al estar también en cero los ocho bits del **MB15.8**, mantendremos el valor cero.

En cambio, si nuestra entrada analógica deja todos los bits valen uno y los ocho bits del **MB15.8** están en cero, el valor obtenido será 255 **(2^8).** Con esto hemos logrado un traspaso de **BYTE a WORD.**

En el **MB15.0** escribo el valor que queremos y en el **MB15.8** nos aseguramos que no hayan más que ceros, por lo tanto, siempre mantendremos el valor de l **MB15.0**.
En el caso de programar un temporizador, debemos decidir el tiempo y esto lo podemos realizar de acuerdo a los conceptos hasta aquí explicados:

L I A 0. 0

Esto significa que estamos asignando una entrada analógica
Luego escribimos:

=MB15.0

Esto quiere decir que el valor de está entrada analógica ha sido asignado a los primeros ocho bits del **MB15.0.** Luego, para asegurarnos que en los ocho bits del **MB15.8** existan solo ceros, asignamos una constante de longitud **BYTE** cero:

LKB0

Una constante de longitud **BYTE** cero, son solamente ceros los asignados a los ocho bits del **MB15.8**. De ésta manera, si a continuación escribimos **MW15** en el temporizador, el valor máximo que podremos obtener será de 255 **(2^8),** lo que significa que el temporizador se det endría en 25,5 segundos **(255 x 0.1).** Recordemos que antes habíamos señalado que el tiempo máximo de temporizador eran 6553,5 segundos, es decir, si usáramos el potenciómetro nos quedaríamos, de no existir ayuda, con un rango de 25,5 segundos, es decir, perderíamos un rango importante del temporizador y, obviamente, esto no es lo que nosotros buscamos.
Para solucionar este problema debemos considerar que el **PS4-111** también puede realizar funciones aritméticas. Estas funciones siempre las realiza en form ato **BYTE**.
Nosotros, por ejemplo, podemos requerir que nuestro temporizador llegue al valor 10000, es decir 1000 segundos. El problema es que nuestro potenciómetro solo me da hasta el 255, por lo tanto, este valor tenemos que multiplicarlo aproximadamente por 40 para llegar a 10000. Esto en la práctica se realiza dela siguiente forma:

L I A 0. 0

Esto significa: “Léeme el valor que tiene el potenciómetro”. Luego escribimos:

M U L K B 40

Esto se interpreta: “Multiplica este valor por la constante de l ongitud BYTE 40”. Luego agregamos:

= M B 15. 0

Esto es equivalente a decir: “Asigna este valor al **MB15.0.**
Ahora bien, si nosotros tenemos el potenciómetro a tope, dispondríamos de un valor de 255. Este valor multiplicado por la constante **BYTE 40** incrementaría el valor a 10000 y algo, lo que significa que esto ya no se puede representar con 8 bits, porque con 8 bits llegamos expresamente al valor 255. Entonces, para todo lo que sobrepasa este valor existe una instrucción llamada **GOR**, que significa recoger el sobrante y escribirlo, es decir:

L I A 0.0
M U L K B 40
= M B 15. 0
G O R = M B 15. 8

Entonces lo que el hace después de la multiplicación, es tomar el valor de menor peso y asignárselo al **MB15.0** y el de mayor peso lo asigna al **MB15.8**. De esta forma tendremos el valor completo de esta multiplicación entre el valor de nuestra entrada analógica y la constante 40.
En el fondo, a través de la instrucción **GOR** ordenamos recoger el valor que sobrepase a **BYTE** y asignárselo al **MB15.8**. Esto significa, que todo valor resultante de esta multiplicación, que no sobrepase el valor 255 será asignado al **MB15.0** y todo lo que sobrepase este valor , será asignado al **MB15.8**.

PRÁCTICA

ELECTRÓNICA

INDUSTRIAL

ENVÍO 12



FICHA PRÁCTICA DE AUTOMATAS PROGRAMABLES

En este capitulo vamos a tratar del software que, como sabemos, se refiere a los programas o partes no tangibles físicamente del autómata. Si bien el software en su amplio término trata tanto de los programas creados por el usua rio como los propios creados por el funcionamiento interno del autómata, nosotros nos vamos a referir a los primeros.

INSTRUCCIONES Y PROGRAMAS.-

Un programa es una sucesión o lista en un determinado orden, de distintas órdenes de trabajo también llamadas instrucciones y capaz de hacer ejecutar al autómata la secuencia de trabajo pretendida.

Una instrucción u orden de trabajo es la parte más pequeña de un programa y consta de dos partes principales: **operación y operando**.

ESTRUCTURA DE UNA INSTRUCCION

En general, toda instrucción está constituida por:

Instrucción = Código de operación + operando

Dirección.- A todas las instrucciones se les asigna un número consecutivo: la dirección. El autómata lleva a cabo las instrucciones en el mismo orden en que están las direcciones.

000
001
002
003
004
005
006

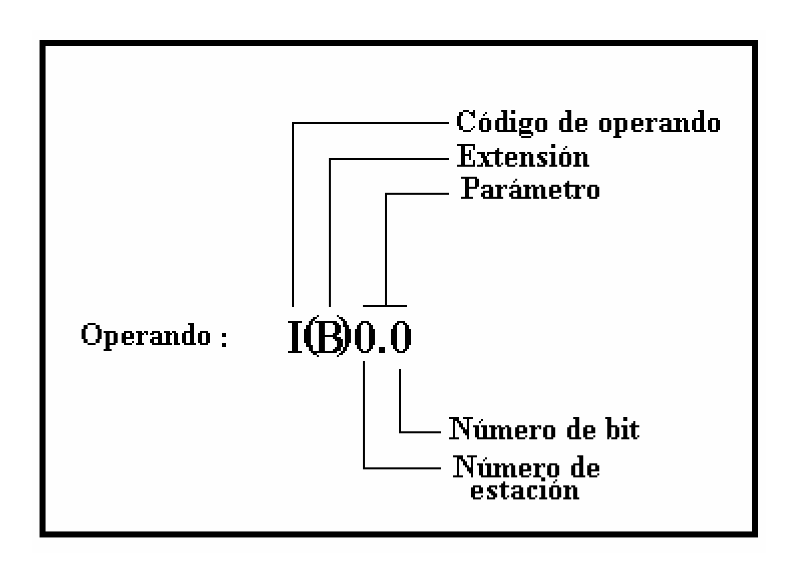
Instrucción. - Cada una de las órdenes de trabajo de un programa se denomina instrucción. Toda instrucción consta de operación y operando.

Operación.- La operación indica lo que se ha de hacer. La operación es el elemento constituyente de una instrucción, que le indica a la CPU qué instrucción ha de ejecutar.

L : Cargar.
A : And.
O : Or
XO : Or excl.
= : Salida o asignación.
S : Activar.
R : Desactivar.

Operando.- El operando indica con que se ha de realizar una operación. El operando consta de un código de operando (+ extensión) y del parámetro del operando.

El operando es el elemento constituyente de una instrucción, el cual le indica a la CPU dónde realizar la instrucción que está en curso.



Código operando

N : Negación.
I : Entrada.
Q : Salida.
M : Merker.
K : Constante.

Extensión.- La extensión del código del operando indica el tipo de datos utilizados:

Sin extensión : Tipo de dato "Bit" por ej. I0
Extensión B : Tipo de dato "BYTE" por ej. IB
Extensión W : Tipo de dato "Palabra" por ej. IW

BIT.- El bit es la unidad de información más pequeña. Puede tener los estados lógicos "1" ó "0" (Conectado o desconectado) y sirve para secuenciar rápidamente entradas y salidas.

BYTE.- El byte consta de 8 bits y por consiguiente puede tener 8 estados "1" ó "0". Se utiliza para procesar valores analógicos. Un BYTE representa los valores decimales entre 0 y 255 (2).

WORD (Palabra).-La palabra consta de 16 bits. Por lo tanto puede tener 16 veces el estado "1" ó "0". Se usa en Módulos o Merker de palabras. Una palabra representa valores decimales entre 0 y 65.535 (2).

Parámetro.- El parámetro consta del número de la estación y del número de bit.

VARIACIONES DE PARÁMETROS EN UNA INSTRUCCIÓN TIPO BIT

DIRECCIÓN

OPERACIÓN

L :Cargar
A :And
O :Or
XO :Or excl.
= :Asignación
S :Activar
R :Desactivar

PARAMETROS

Entradas I : 0.........15
Salidads Q : 0.........15
Merquer M
No aplicable con constantes

0 0 0 L I 0. 1 LDI

CON UNIDADES I y Q

0 :Aparato base
1 :1° Extención
2 :2° Extención
3 :3° Extención

CON MERQUERS M

0 :En palabra de Merquer 0
1 :En palabra de Merquer 1
.
.
35 :En palabra de Merquer 35

Con constantes K 0 ó 1

OPERANDO

N :Negación
I :Entrada
Q :Salida
M :Merquer
K :Constante

VARIACIONES DE PARÁMETROS EN UNA INSTRUCCIÓN TIPO BYTE

DIRECCIÓN

OPERACIÓN

L :Cargar
A :And
O :Or
XO :Or excl.
= :Asignación

PARAMETROS

Entradas IB :0 / 8
Salidas QB :0 / 8
I Analógicas (IA) :0 3
Q Analógicas (QA) : 0
Merquers (Mb) :0 / 8
No aplicable con constantes

0 0 0   O I 0. 8   LDI

CON UNIDADES I y Q

0 :Aparato base
1 :1° Extención
2 :2° Extención
3 :3° Extención

CON MERQUERS M

0 :En palabra de Merquer 0
1 :En palabra de Merquer 1
.
.
35 :En palabra de Merquer 35

Con constantes KB

0.................255

OPERANDO

IB :Entrada Byte
QB :Salida Byte
MB :Merquer byte
IA :Entrada analóga
QA :Salida análoga
KB : Constante bite.

VARIACIONES DE PARÁMETROS EN UNA INSTRUCCIÓN TIPO PALABRA

DIRECCIÓN

OPERACIÓN

L :Cargar
= :Asignación

0 0 0   L I W 1   LDI

OPERANDO

IW :Entrada Word
QW :Salida Word
MW :Merquer Word
KW :Constante Word

CON UNIDADES I y Q

0 :Aparato base
1 :1° Extención
2 :2° Extención
3 :3° Extención

CON MERQUERS M

0 :En palabra de Merquer 0
1 :En palabra de Merquer 1
.
.
35 :En palabra de Merquer 35

Con constantes KW

0..........65 535

SISTEMAS O LENGUAJES DE PROGRAMACIÓN. -

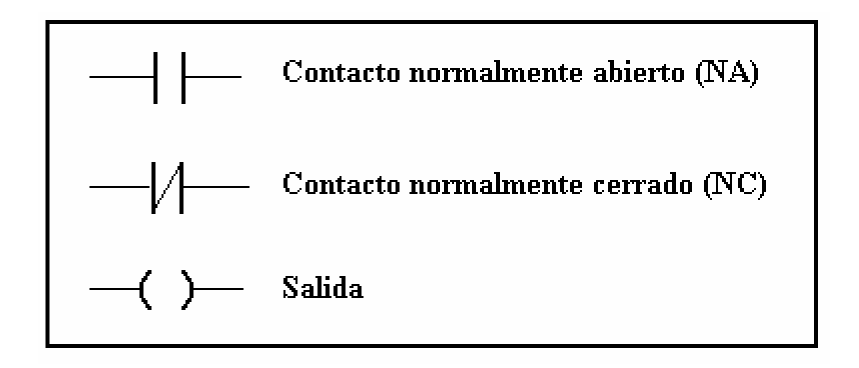
Varios son los sistemas o lenguajes de programación posibles en los Autómatas Programables, aunque su utilización no se puede dar en todos los autómatas; es por esto que cada fabricante indica en las características generales de su equipo, el lenguaje o los lenguajes con los que puede operar. En general, se podría decir que los lenguajes de programación más usuales son aquellos que se enumeran a continuación:

a)Lista de instrucciones o de abreviaturas nemotécnicas. Se define como nemotécnico, porque corresponde a la abreviatura o sigla de una instrucción del programa, que define de una forma aproximada la operación que realiza. Ejemplo: LD para LOAD.

LI0.0
ANI0.1
OI0.2
=Q0.0
EP

b)Esquema de contactos (Ladder o diagráma de escalera).- Los esquemas de contactos o diagrama de escalera utilizan símbolos de contactos normalmente abiertos o normalmente cerrados. La figura siguiente muestra un **RUNG** (peldaño) de un esquema de contactos. Un **RUNG** es la simbología de contactos necesaria para controlar una salida.

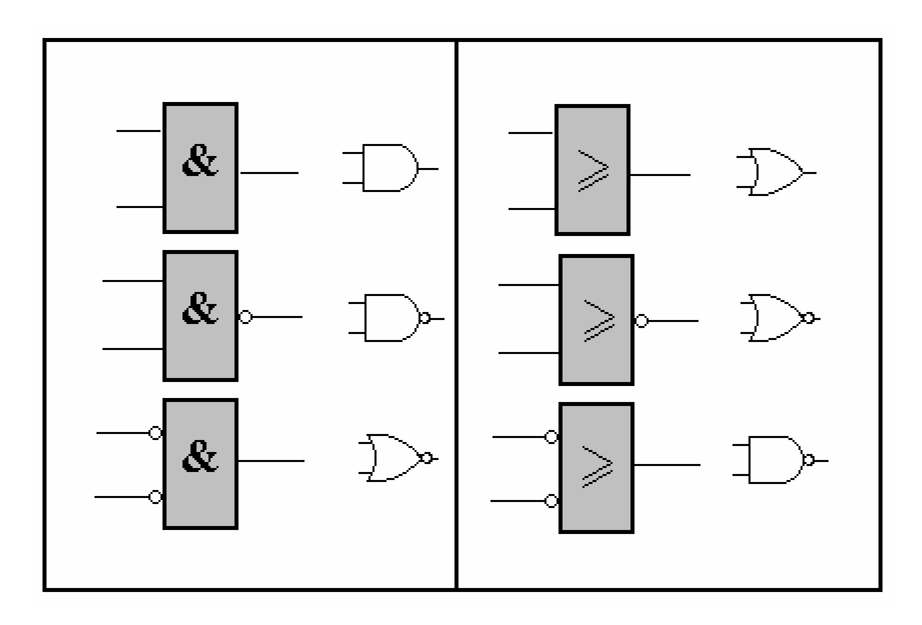
Los siguientes símbolos son utilizados para contactos:

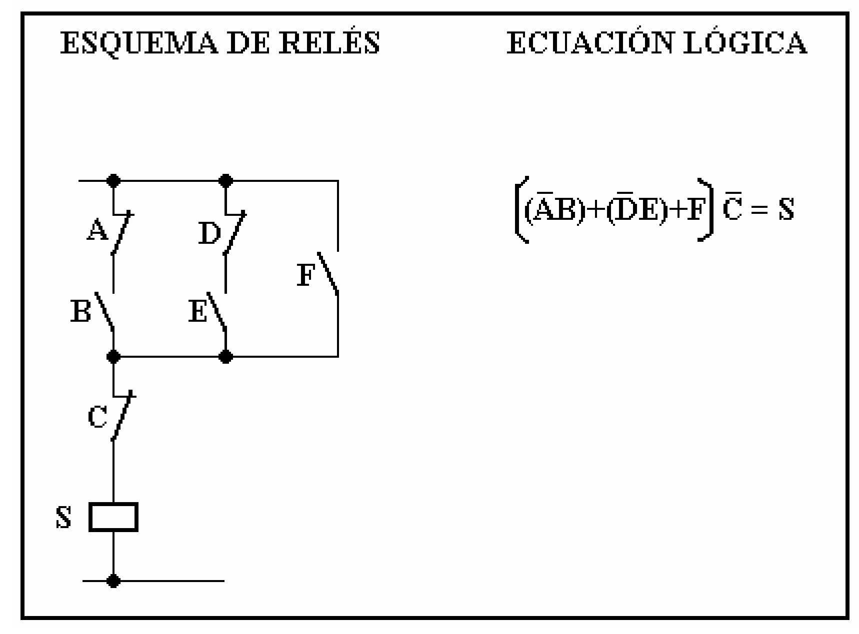


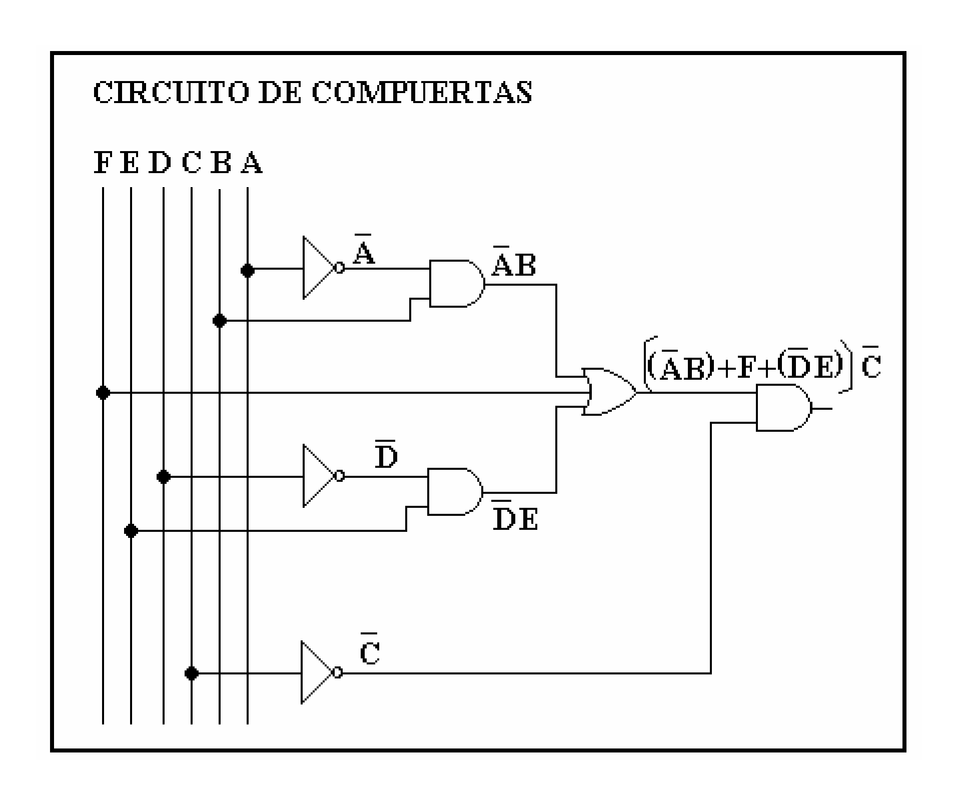
c)Esquema de bloques funcionales.

Excepto el nemónico, los demás tienen como base su repr esentación gráfica, pero todos ellos deben ir acompañados de la relación de líneas de programa que configuran el mismo.

Para entender mejor estos lenguajes, a continuación se realiza una explicación de ellos. En el caso de los tres primeros, que son los má s utilizados, se ha puesto un ejemplo de cada uno tomando como base un mismo circuito y partiendo de la ecuación lógica del mismo, de su esquema de relés y del circuito con puertas lógicas.



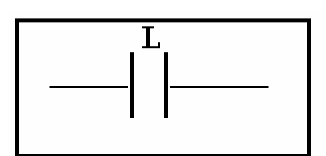




Lista de instrucciones (nemónicos).-

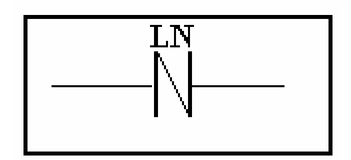
Es un lenguaje en el cual cada instrucción es representada a través de una sigla. A continuación figura una relación de nemónicos, con indicación de lo que representan:

L (Load) Carga. En una secuencia de un programa, siempre la primera línea de un contacto normal abierto, se inicia con la instrucción (L). Cuando una línea lógica comienza con un contacto N.A., esta instrucción indica que se comienza en la dirección especificada una nueva línea o sublínea.

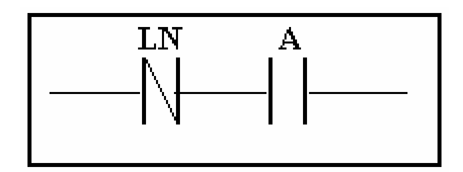


LN (Load Not) : En una lista de instrucciones, cada línea con un primer contacto normal cerrado, comienza con la instrucción LN.

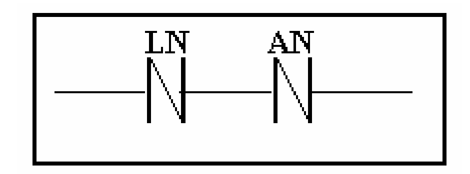
Se utiliza esta instrucción en lugar de LD, cuando una línea lógi ca empieza con un contacto N.C..



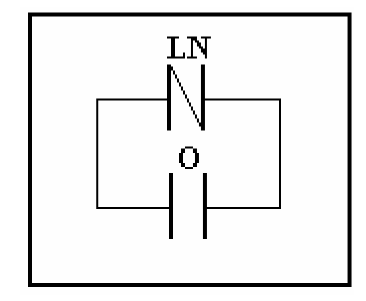
A (And): En una lista de instrucciones o en un esquema de contactos esta instrucción coloca un contacto normal abier to en serie con lo definido anteriormente. Esto significa que realiza la operación lógica AND de dos o más contactos, es decir conecta dos o más contactos en serie.



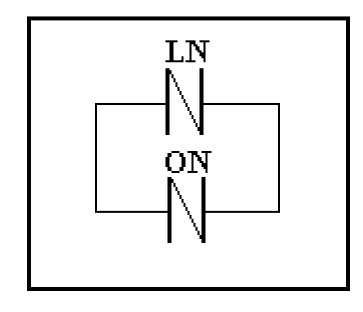
AN (and Not): En una lista de instrucciones o en un esquema de contactos ésta instrucción coloca un contacto normal cerrado en serie con lo definido anteriormente. Es decir, conecta en serie contactos N.C.



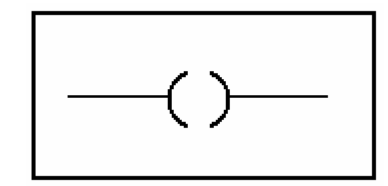
O (OR) : En una lista de instrucciones o en un esquema de contactos esta instrucción coloca un contacto normal abierto en paralelo con lo definido anteriormente. Es decir, conecta en paralelo dos o más contactos



<u>ON (Or Not)</u>: En un esquema de contactos o en una lista de instrucciones esta instrucción coloca un contacto normal cerrado en paralelo con lo definido anteriormente. Es decir, conecta en paralelo contactos N.C.



<u>= Q (Equal)</u> : Bobina de relé de salida.



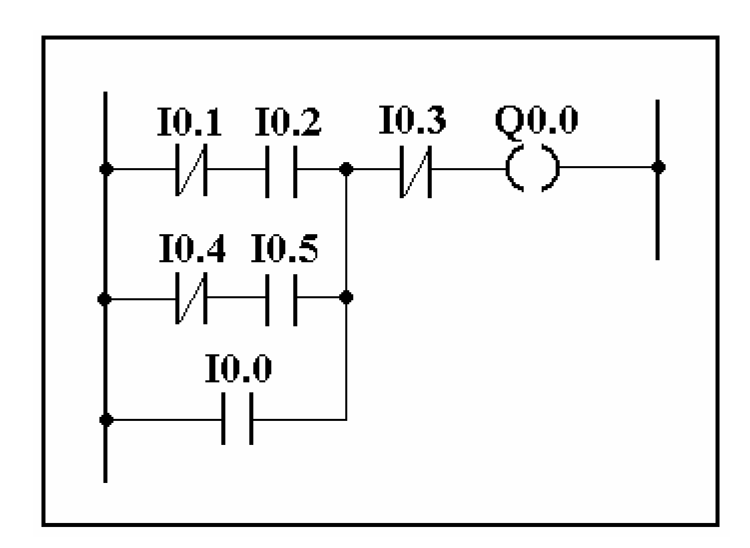
<u>EP (End Program)</u> : Término de programa. Debe colocarse en los bloques finales del programa.

DESARROLLO PRÁCTICO CON CONTACTOS

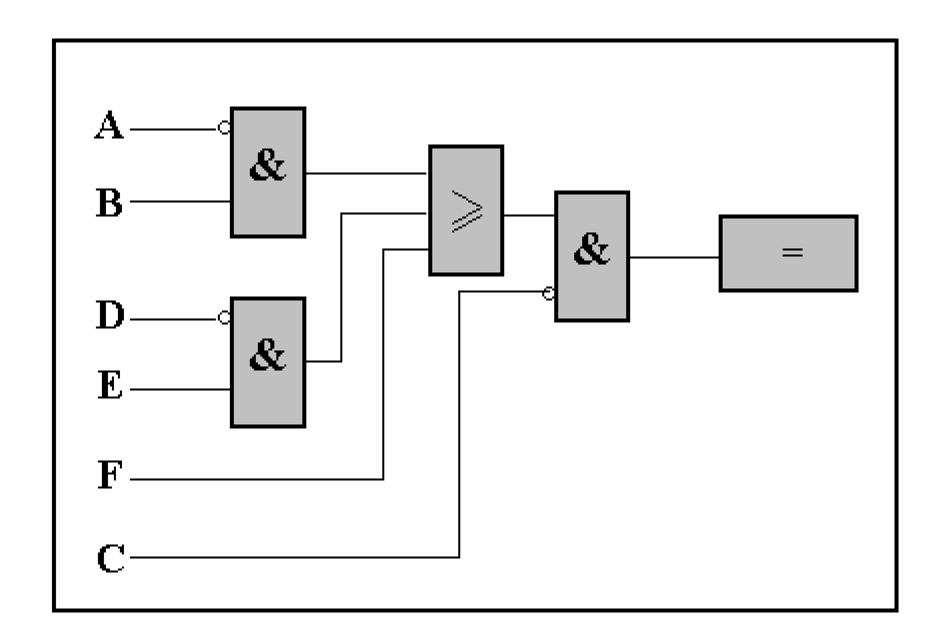
Observe la siguiente lista de instrucciones y de acuerdo a ella desarrolle los correspondientes esquemas de contactos y esquema de bloques funcionales:

```
LNI0.1
AI0.2
LNI0.4
AI0.5
OI0.0
O
ANI0.3
=Q0.0
EP
```

El esquema de contactos correspondiente a este circuito es el siguiente:



A través de esta lista de instrucciones también se logra el siguiente esquema de bloques funcionales:

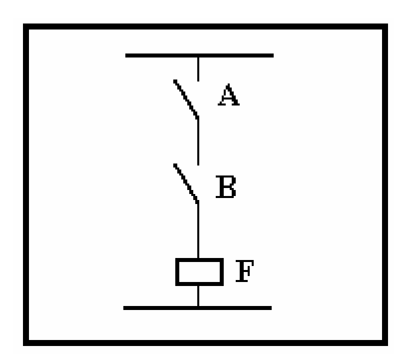


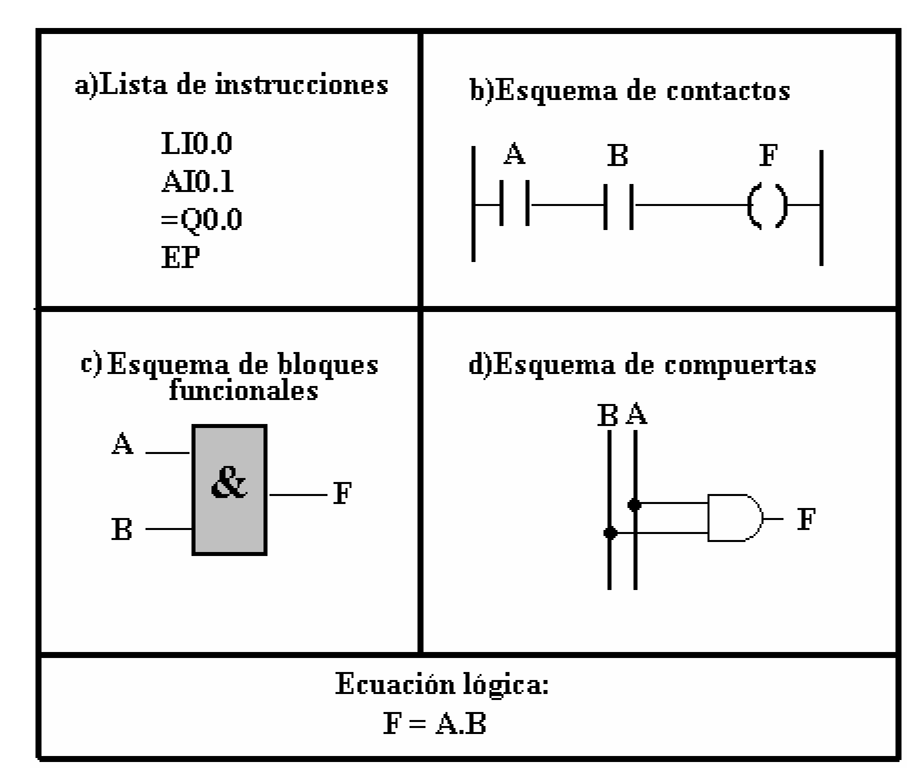
DESARROLLAR LOS SIGUIENTES EJERCICIOS CON SECUENCIAS TIPO BIT:

Para las funciones lógicas enumeradas a continuación, se pide desarrollar:

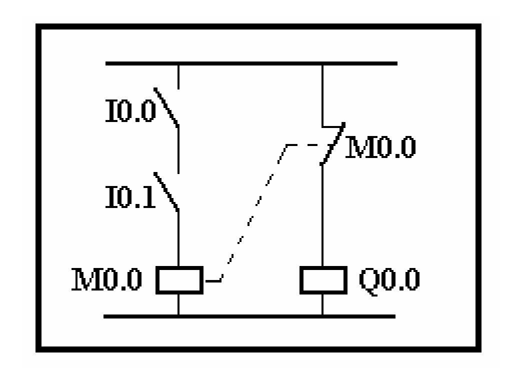
a)Lista de instrucciones.
b)Esquema de contactos.
c)Esquema de bloques funcionales.
d)Ecuación lógica.
e)Esquema de compuertas.

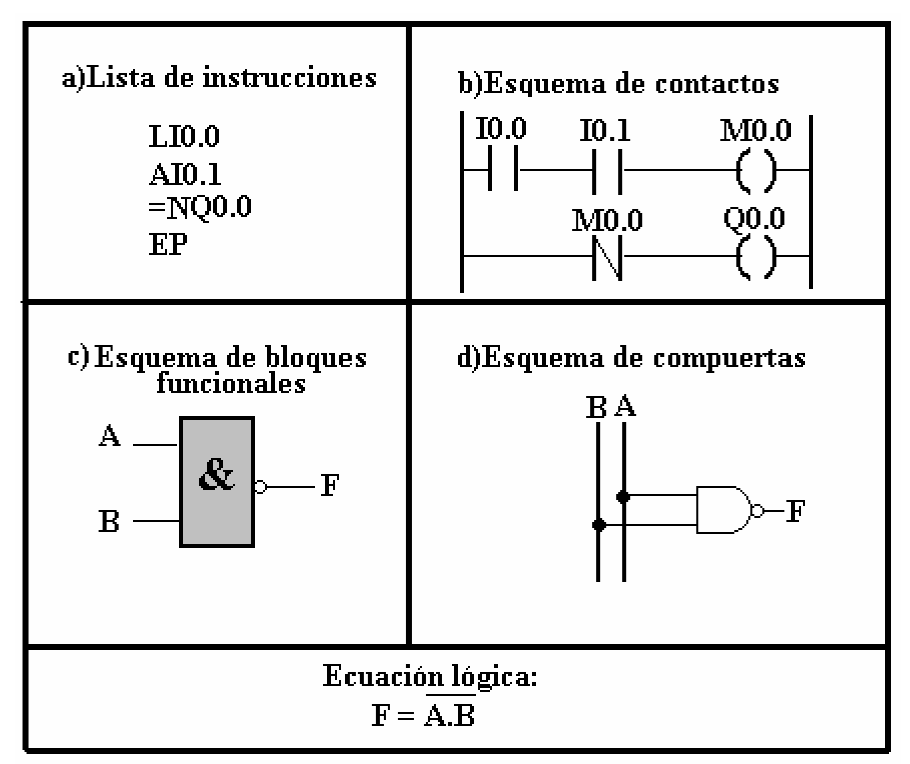
1.-FUNCION LÓGICA AND.- La salida de una puerta AND estará a "1" si todas las entradas están a "1".



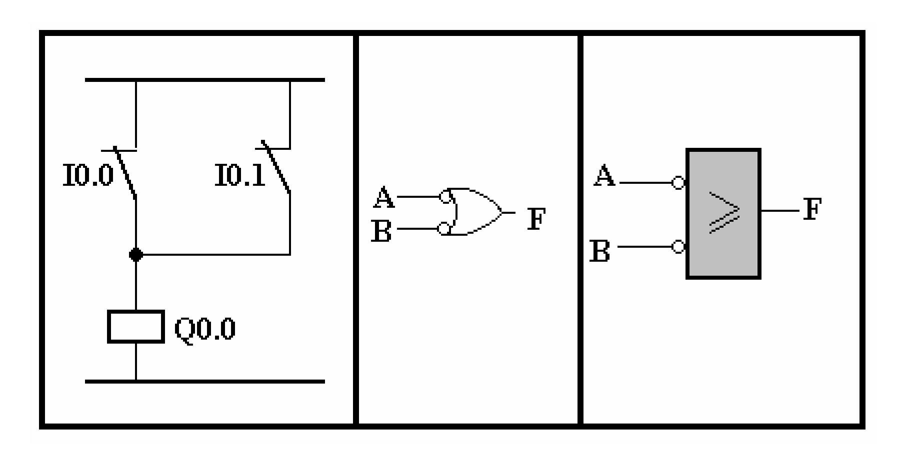


FUNCIÓN LÓGICA NAND.- La salida estará a "0" cuando todas las entradas esten a "1".

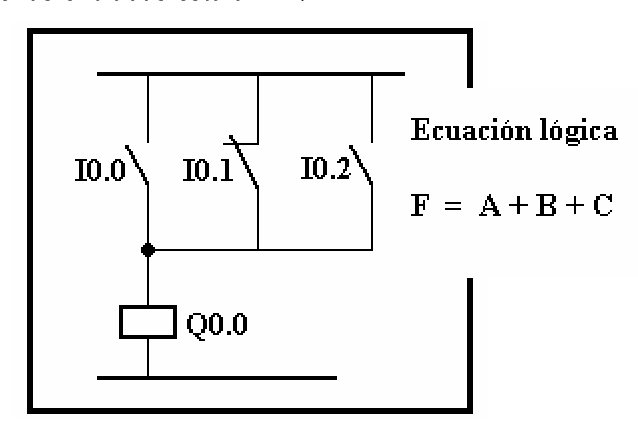


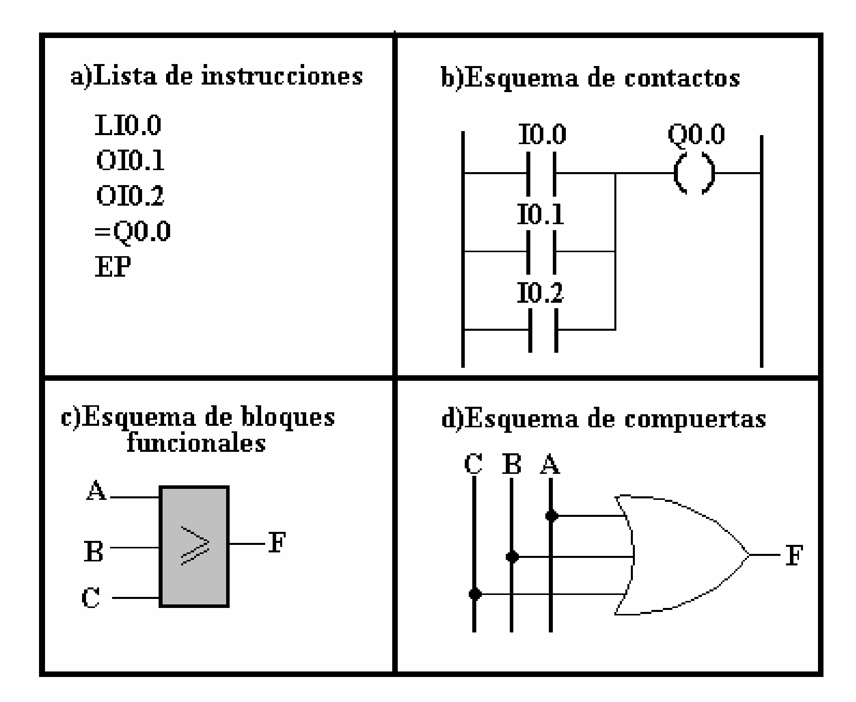


Una puerta NAND funciona del mismo modo que una puerta OR con entradas negadas. Una salida asignada no necesita ser negada

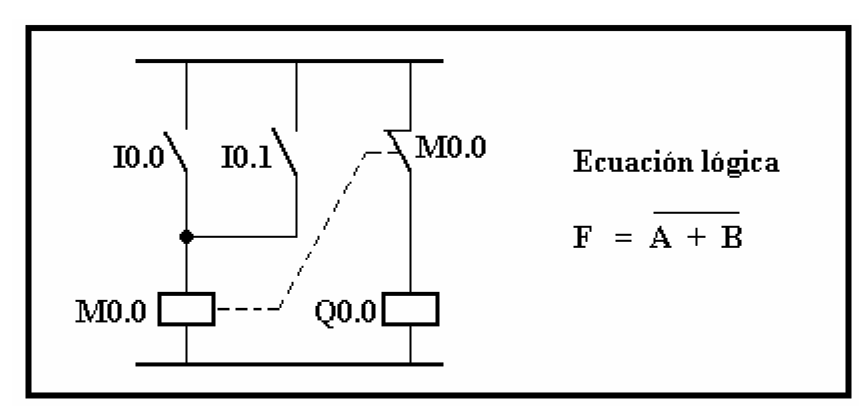


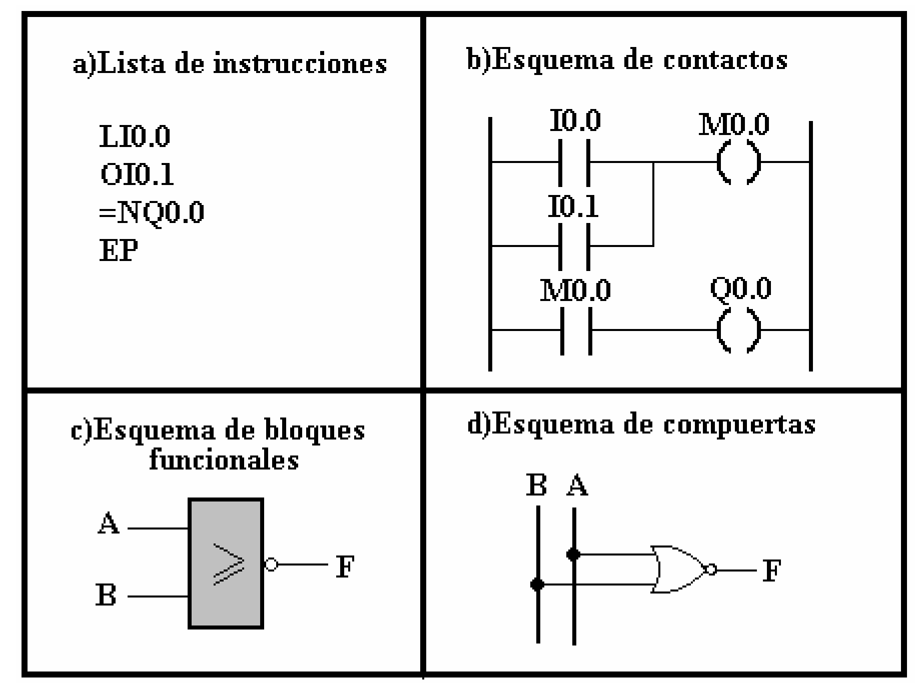
FUNCIÓN LÓGICA OR.- La salida de una puerta "OR" está a "1" cuan do por lo menos una de las entradas está a "1".



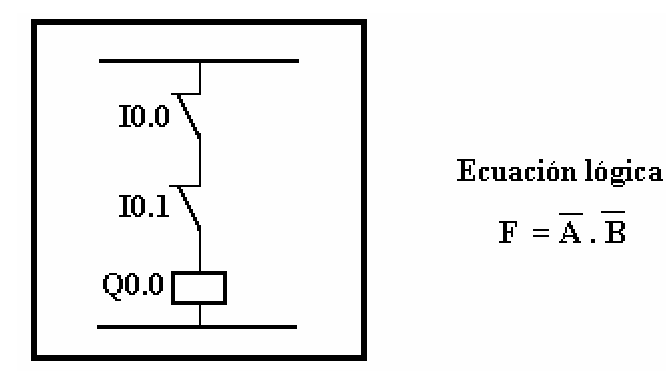


FUNCIÓN LÓGICA NOR (OR NOT).- La salida está a "0" cuando por lo menos una de las entradas está a "1".



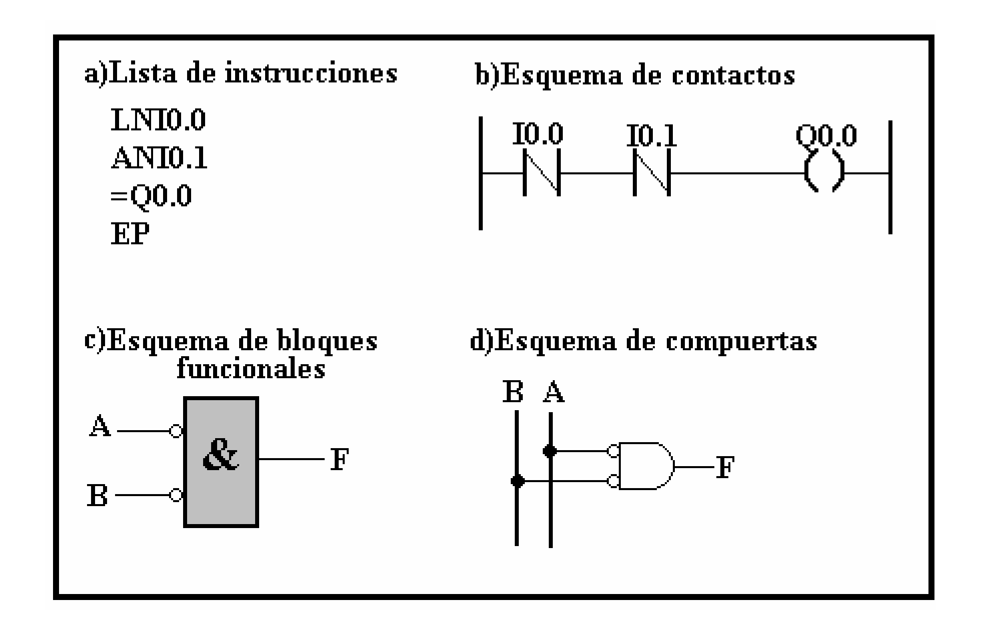


Una puerta NOR funciona del mismo modo que una compuerta AND con entradas negadas. Una salida Q asignada no necesita ser negada.

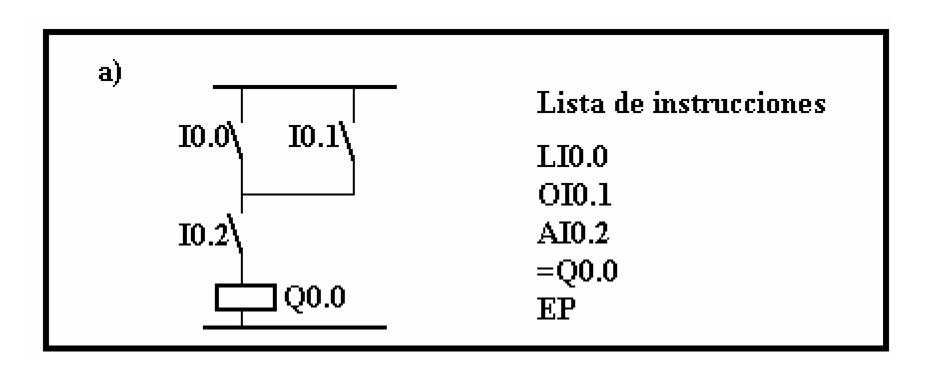


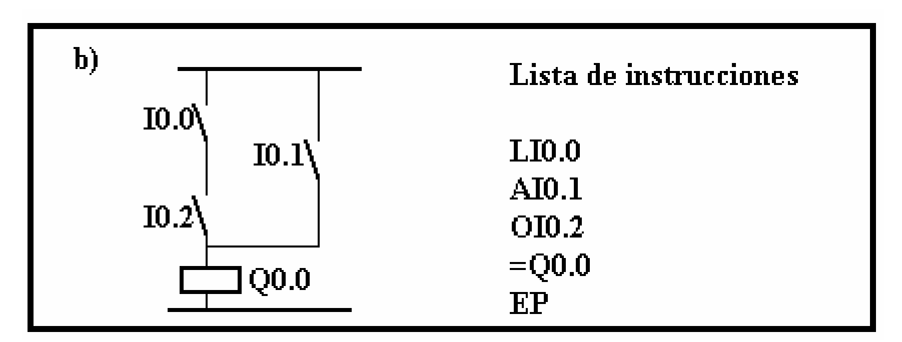
Ecuación lógica

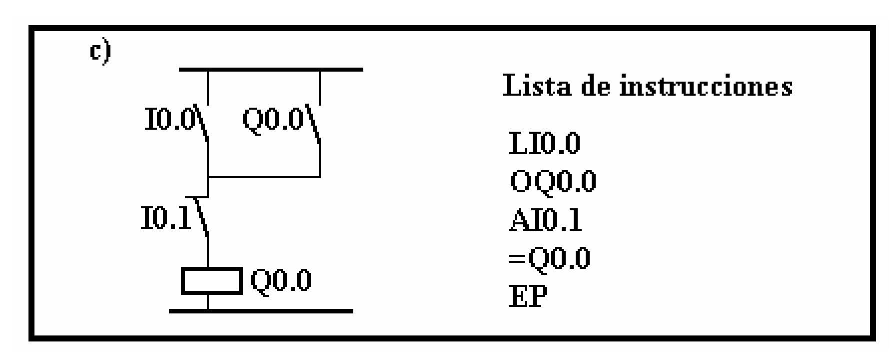
F = $\overline{A}.\overline{B}$

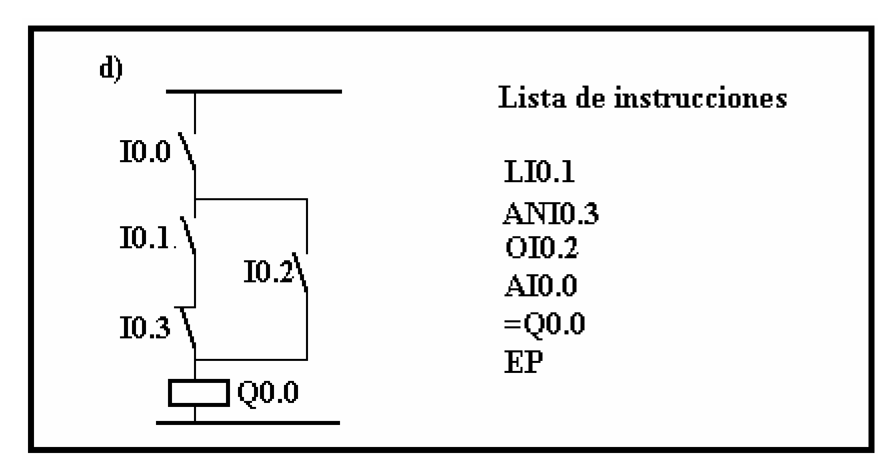


CIRCUITOS COMBINADOS AND / OR.-





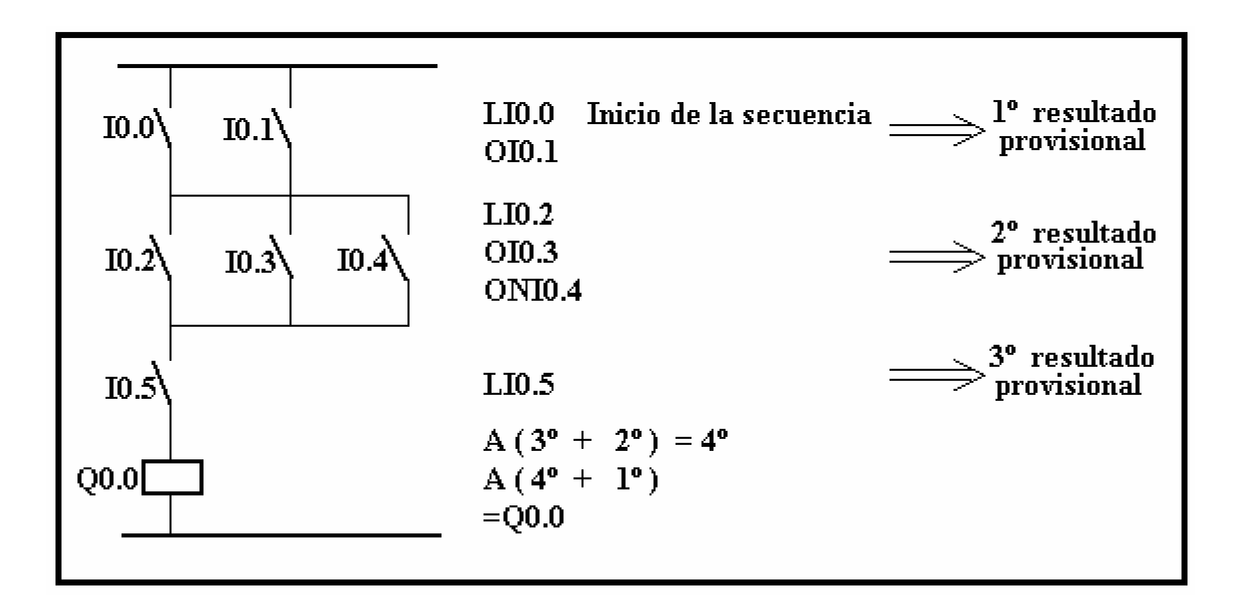




REGISTRO DE PILAS (STACK REGISTER). -

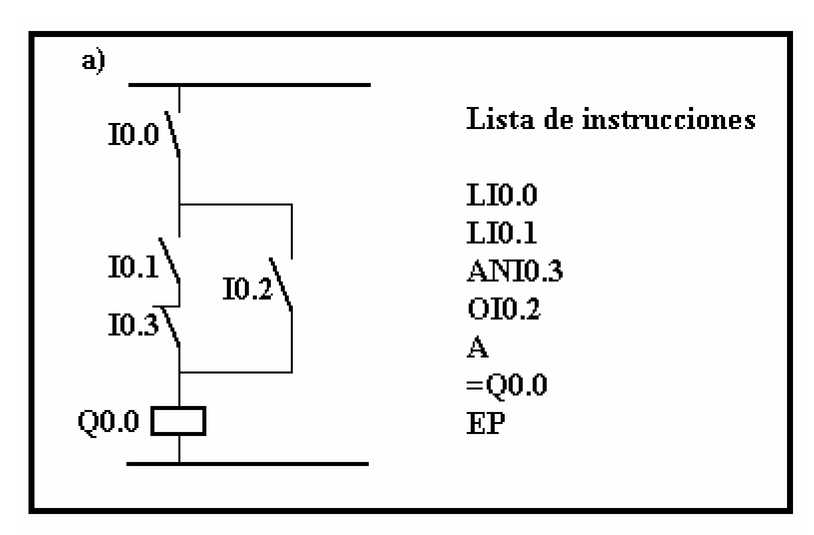
El PS4 - 111 puede procesar hasta 8 resultados provisionales por secuencia, no requiriendo Merkers provisionales, sino que son almacenados en un registro LIFO (Last in first out).

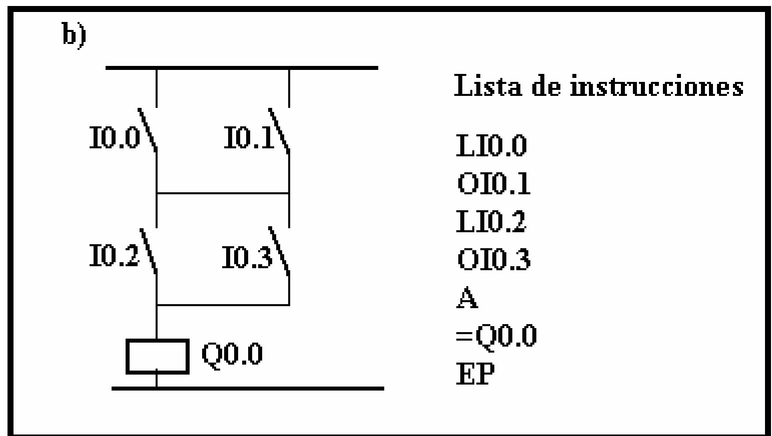
Los siguientes contactos han de secuenciarse:

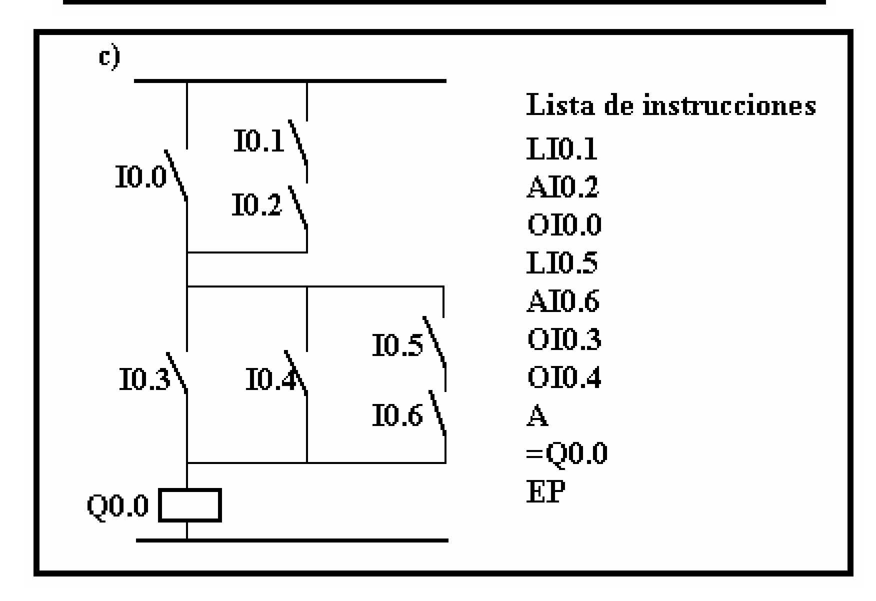


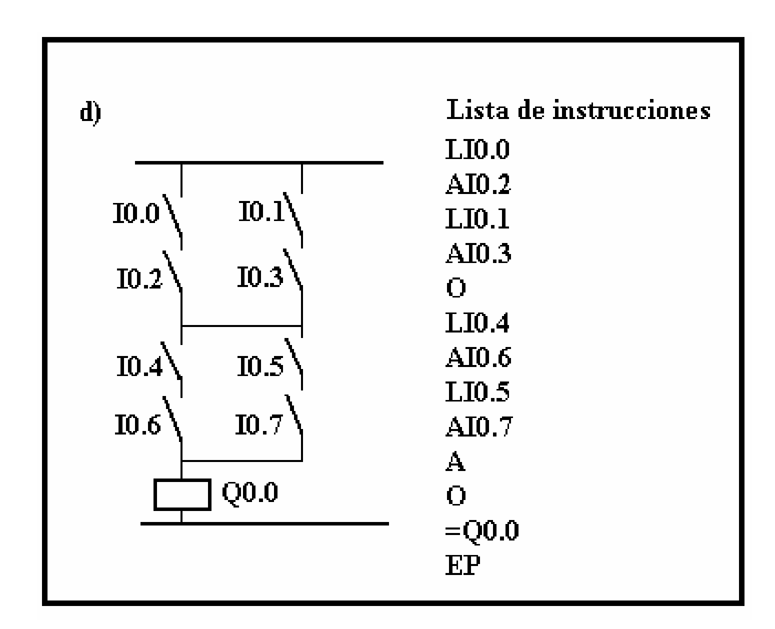
EJEMPLOS DE CIRCUITOS COMBINADOS AND / OR .

La solución se obtiene con ayuda del registro de pila (Stack).









<u>CONSIDERACIONES PREVIAS A LA PROGRAMACIÓN:</u>

Antes de acometer los ejemplos prácticos, es necesario tener en cuenta algunas consideraciones que nos facilitarán la labor de programación y que son las siguientes:

a)La programación de cada bloque de contactos se realizará en el orden de izquierda a derecha.

b)El sentido de programación de los bloques de contactos de un programa se ejecutarán en el sentido de arriba hacia abajo.

c)El número de contactos que se pueden colocar en un bloque, desde el comienzo de la línea principal hasta la salida, es ilimitado. La única limitación práctica que podemos encontrarnos es la anchura del papel cuando queramos sacar el programa por impresora; en este caso, el número máximo de contactos en serie es de ocho.

d)Al no existir limitación de contactos, es preferible realizar un circuito claro y comprensible con un elevado número de contactos que uno complicado como consecuencia de reducir el número de éstos.

e)No se puede conectar una salida directamente a la línea principal, en estos casos se intercala un contacto cerrado en serie con ella.

f)Después de una salida no se puede colocar contacto alguno.

g)En nuestro autómata es posible programar dos salidas. en paralelo.

h)Los términos contacto abierto, normalmente abierto (N.A.) o contacto de cierre, significan lo mismo, es decir que el paso de corriente a través de él no es posible.

En el mismo sentido, el término contacto cerrado, normalmente cerrado (N.C.) y contacto de apertura también significan lo mismo y es el contacto que en estado de reposo se encuentra cerrado, o sea, el paso de corriente a través de él sí es posible.

i)Contactos de entrada. El número de contactos abiertos o cerrados que se pueden utilizaren un programa, por cada uno de las entradas es ilimitado, o sea, se puede repetir el mismo número de contacto cuantas veces queramos y tanto abierto como cerrado.

j)Contactos de salida. El número de salidas o bobinas de salida o relés de salida es fijo, por lo que no se puede repetir un mismo número de salida, pero, por el contrario, el número de contactos asociados a cada una de ellas, tanto abiertos como cerrados es, al igual que en el caso anterior, ilimitado.

k)Contactos de marcas (Merkers). Aunque no son salidas exteriores, los merkers se representan y programan de forma similar, siendo su utilización más común como relés auxiliares.

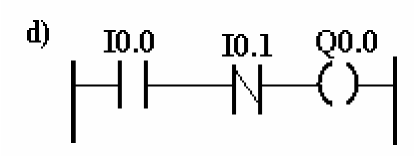
DESARROLLE LOS SIGUIENTES EJERCICIOS

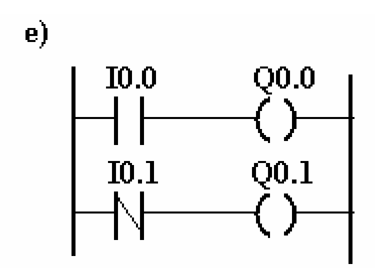
Señale la ecuación lógica y la lista de instrucciones de los siguientes esquemas de contactos:

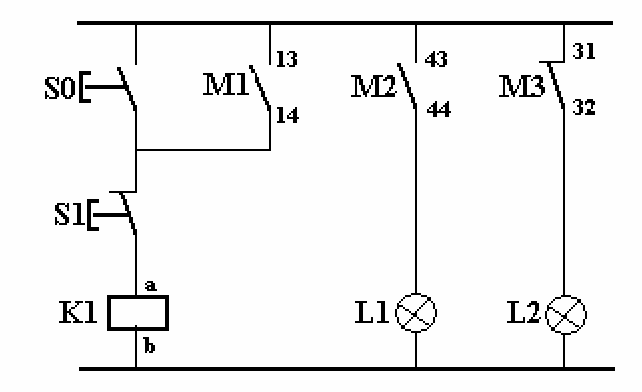
a) I0.0 Q0.0

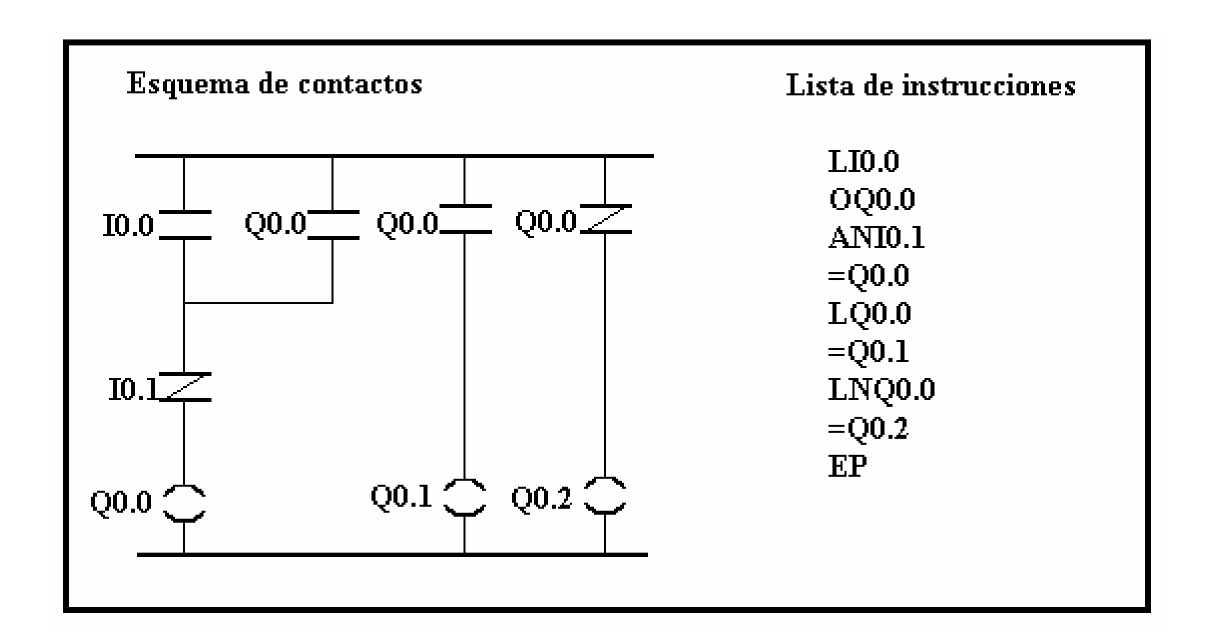
b) I0.0 I0.2 Q0.0
I0.1

c) I0.0 Q0.0
I0.1


d)
I0.0
I0.1
Q0.0

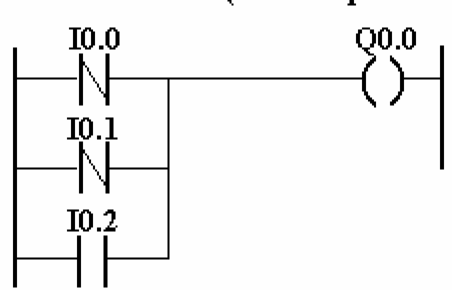

e)
I0.0
Q0.0
I0.1
Q0.1


PARTIDA Y PARADA DE UN MOTOR
S0
M1
13
14
M2
43
44
M3
31
32
S1
K1
a
b
L1
L2
Donde:
I1 e I2 son entradas externas
K1 es salida externa
M1, M2 Y M3 son relés internos (Merkers)

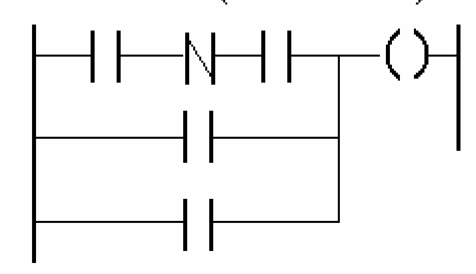


DESARROLLAR EN LISTA DE INSTRUCCIONES LOS SIGUIENTES ESQUEMAS DE CONTACTO

EJEMPLO Nº1 (Circuito serie)



EJEMPLO Nº3 (Circuito mixto)



EJEMPLO Nº4 (Ciecuito mixto)

I0.0 I0.1 I0.2 I0.3 I0.4 Q0.0

I0.5

EJEMPLO Nº5 (Circuito mixto)

I0.0 I0.1 I0.2 I0.8 Q0.0

I0.3

I0.4 I0.5 I0.6 I0.7

EJEMPLO Nº6 (Circuito mixto)

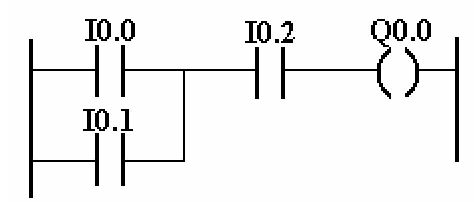
I0.0 I0.1 I0.2 I0.6 Q0.0

I0.3

I0.4 I0.5

I0.7

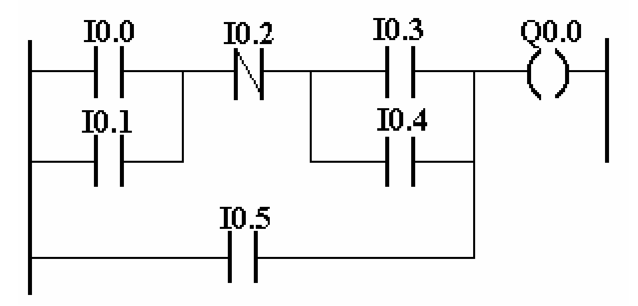
EJEMPLO Nº7 (Circuito mixto)



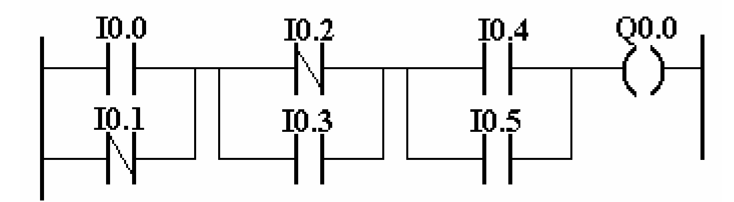
EJEMPLO N°8 (Circuito mixto)

I0.0 I0.1 I0.3 I0.4 I0.5 I0.6 Q0.0
I0.2
I0.7
I0.8 I0.9

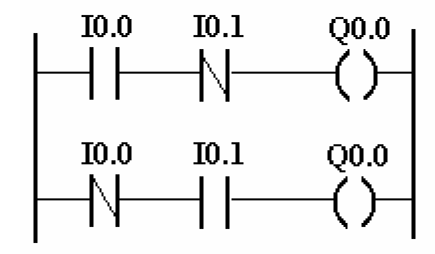
EJEMPLO N°9 (Circuito mixto)



EJEMPLO N°10 (Circuito mixto)

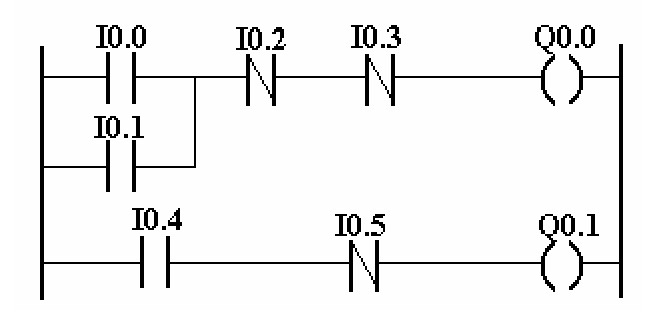


EJEMPLO N°11 (Circuito mixto)



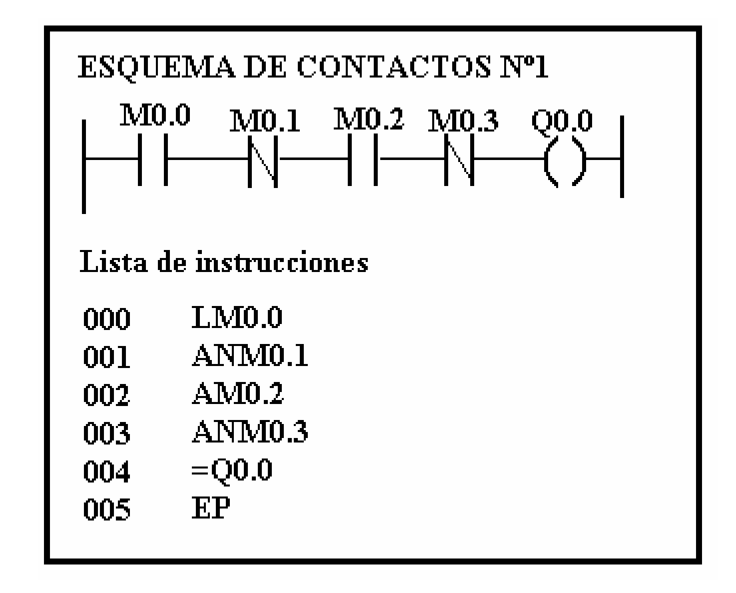
EJEMPLO Nº12 (Circuito mixto)

I0.0 I0.1 I0.2 I0.3 I0.4 I0.5 I0.6 I0.7 Q0.0



DESARROLLOS PRACTICOS CON ESQUEMAS DE CONTACTOS

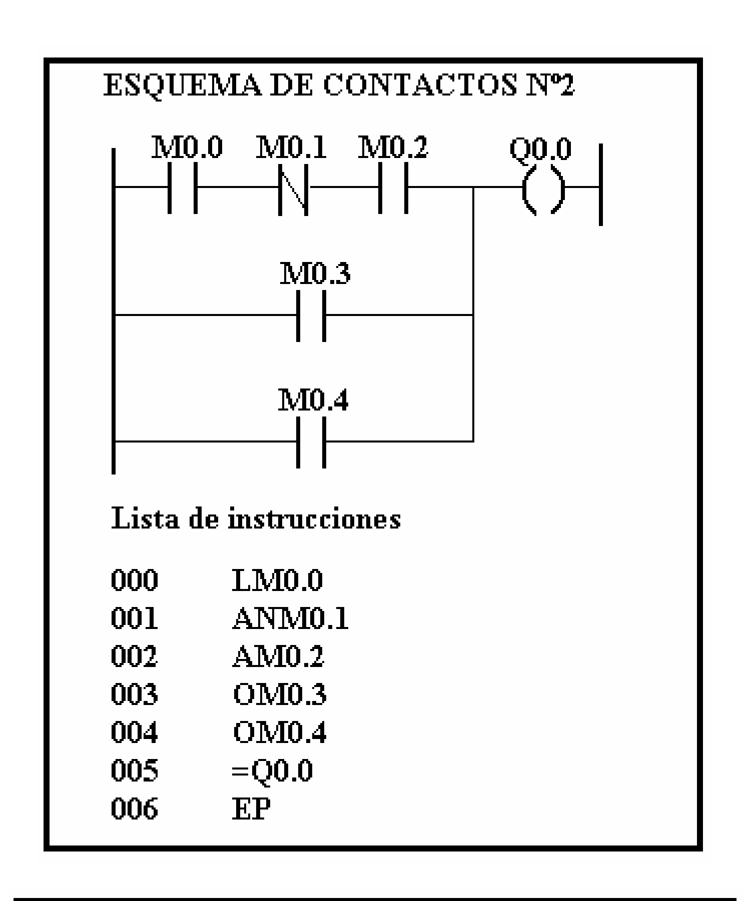
Exprese a través de una lista de instrucciones cada uno de los esquemas de contacto que se muestran a continuación:



ESQUEMA DE CONTACTOS Nº1

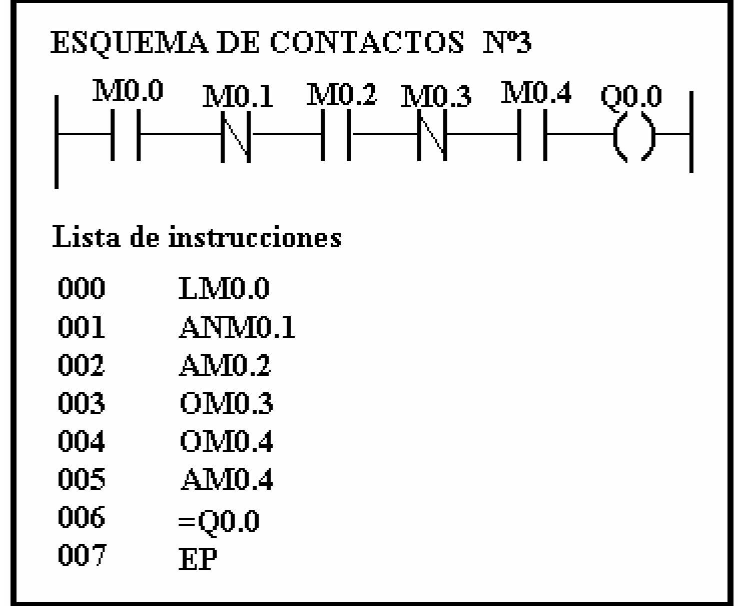
Lista de instrucciones

000	LM0.0
001	ANM0.1
002	AM0.2
003	ANM0.3
004	=Q0.0
005	EP



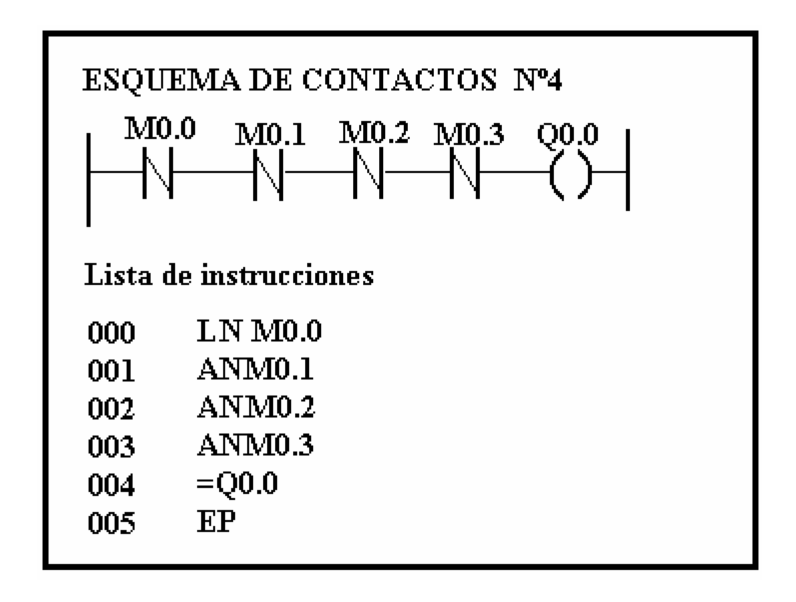
Lista de instrucciones

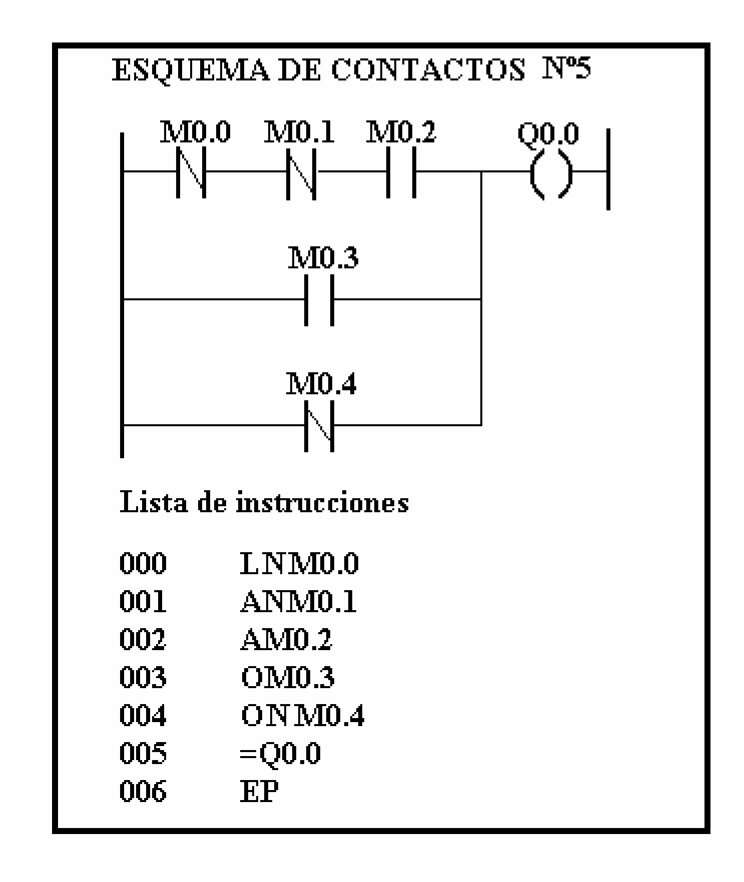
000	LM0.0
001	ANM0.1
002	AM0.2
003	OM0.3
004	OM0.4
005	=Q0.0
006	EP

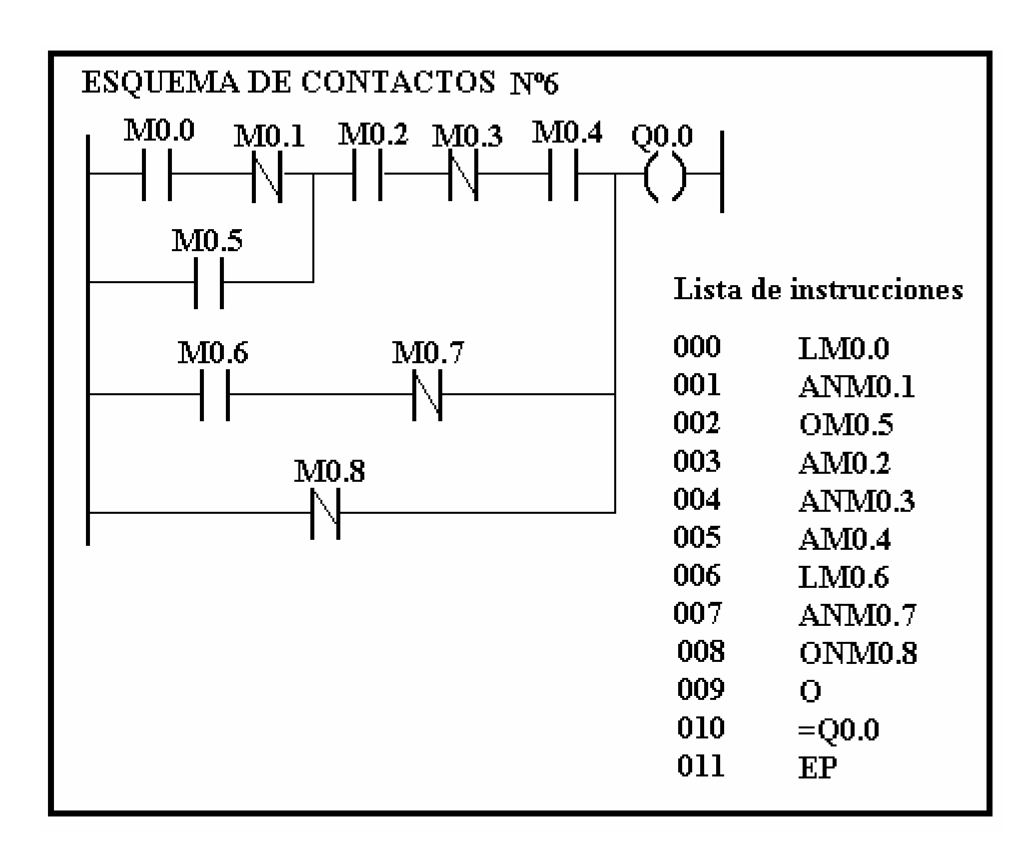


Lista de instrucciones

000	LM0.0
001	ANM0.1
002	AM0.2
003	OM0.3
004	OM0.4
005	AM0.4
006	=Q0.0
007	EP


ESQUEMA DE CONTACTOS Nº4
M0.0
M0.1
M0.2
M0.3
Q0.0
Lista de instrucciones
000 LN M0.0
001 ANM0.1
002 ANM0.2
003 ANM0.3
004 =Q0.0
005 EP


ESQUEMA DE CONTACTOS Nº5
M0.0
M0.1
M0.2
Q0.0
M0.3
M0.4
Lista de instrucciones
000 LNM0.0
001 ANM0.1
002 AM0.2
003 OM0.3
004 ONM0.4
005 =Q0.0
006 EP



TEMPORIZADORES (TR)

Si bien esta instrucción no realiza una operación lógica la incluiremos dentro de este grupo por ser de gran uso en los diagramas de contactos o en las listas de instrucciones

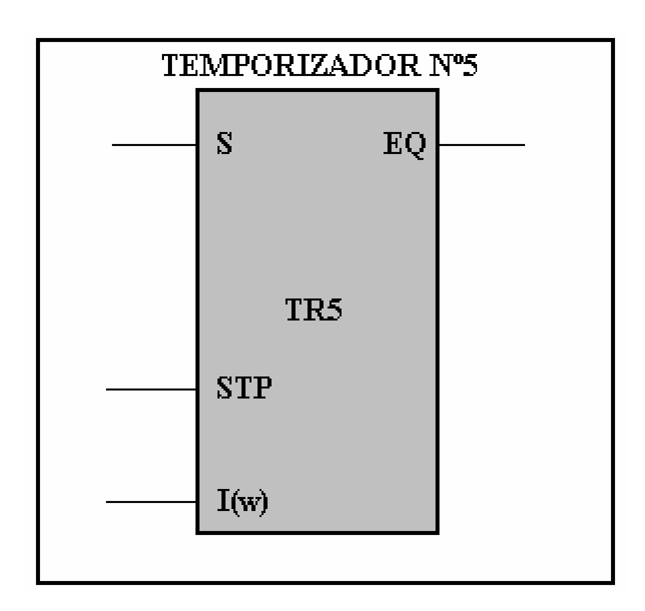
El PS4 - 111 incluye las posibilidades de programar hasta 32 temporizadores, con márgenes de retardo de 0,1 a 6553,5 segundos. .

TR0 ....31 son temporizadores con retardo a la llamada con una entrada de stop. Cuando hay un flanco ascendente en la entrada “S”, el valor de retardo presente en “I” es aceptado. La salida “EQ”se pone en alto después del tiempo “t” .El tiempo de retardo es el resultado de:

t = factor de retardo x 0.1 s

El factor de retardo puede variar entre 1 y 65535. Cua ndo la entrada de stop está en alto el tiempo puede interrumpirse, es decir, el tiempo de retardo “t” se alarga en el tiempo “ts” durante el cual la entrada de stop está en alto. El módulo se pone a cero cuando la señal en la entrada “S” pasa de alto a bajo.

El diagrama en bloques del temporizador es:



FICHA PRÁCTICA DE TEMPORIZADORES

Los temporizadores son instrucciones de salida que son usadas para activar o desactivar dispositivos después de un intervalo de tiempo. **Los temporizadores** generalmente son consideradas salidas internas.

La figura anterior muestra el diagrama en bloques de un temporizador.

Podemos notar que nuestro temporizador posee tres entradas que son :

a)Set (S) .... Entrada de arranque.
b)Stop. (Stop) .... Entrada de stop.
c)Input. Word (I). .... Margen de tiempo.

La salida del temporizador es:

Equal (EQ) .... Salida.

NOTA: En la entrada Input Word (IW) podemos programar un valor de tiempo fijo o un valor de tiempo variable, por lo que será bastante importante dar a conocer las instrucciones a emplear para cada caso:

PARA PROGRAMAR TIEMPOS FIJOS

En la entradas Input Word debemos escribir : **K (constante) - W (longitud palabra) - tiempo en segundos multiplicado por 10.**

Ejemplo Nº1: Se requiere un temporizador con retardo a la conexión con selección de tiempo constante de 5 segundos y sin entrada stop.

```
000      TR0

         SET :   I0.0
         STP :
         IW  :   KW50
         EQ  :   Q0.0
```

Ejemplo Nº2: Se requiere un temporizador con retardo a la desconexión, con selección de tiempo constante de 20 segundos y sin entrada s top.

```
000      TR0

         SET :   I0.0
         STP :
         IW  :   KW200
         EQ  :   NQ0.0
```

Ejemplo Nº3: Se requiere un temporizador con retardo a la conexión, con entradas estáticas para arranque y stop y con selección de tiempo constante de 15 segundos.

```
000      TR0

         SET :   I0.0
         STP :
         IW  :   KW150
         EQ  :   Q0.0
```

Ejemplo Nº4: Generador de impulsos con temporizador (Flip - flops).

```
000        TR0

           SET :   NM0.0
           STP :
           IW  :   KW150
           EQ  :   Q0.0

001        TR1

           SET :   Q0.0
           STP :
           IW  :   KW200
           EQ  :   M0.0
```

PARA PROGRAMAR TIEMPOS VARIABLES DE HASTA 25,5 SEGUNDOS MÁXIMO

```
000        LIA0.0
001        =MB15.0
002        LKB0
003        =MB15.8
004        TR0

           SET :   I0.0
           STP :
           IW  :   MW15.0
           EQ  :   M0.0
```

PARA PROGRAMAR TIEMPOS VARIABLES SUPERIORES A 25,5 SEGUNDOS

a)Ejemplo para programar de 0 a 1000 segundos:

```
LIA0.0
MULKB40
=MB15.0
GOR
=MB15.8
TR0

S    :   I0.0
STP  :
IW   :   MW15.0
EQ   :   M0.0
EP
```

b)Temporizador con un tiempo máximo de 6502,5 segundos :

```
000     TR0

        S    :   I0.0
        STP  :
        IW   :   MW15.0
        EQ   :   M0.0

001     LIA0.0
002     MULKB255
003     =MB5.0
004     GOR
005     =MB5.8
006     EP
```

Los siguientes circuitos corresponden a posibles aplicaciones de un temporizador:

1.- (PORTÓN) ACTIVACIÓN DE UN PORTÓN DE CORREDERA A TRAVÉS DE IMPULSO DE 0,5 SEG. :

LISTA DE INSTRUCCIONES

```
000        LNI0.0
001        ANI0.1
002        LI0.2
003        OQ0.0
004        ANM0.0
005        ANQ0.2
006        A
007        =Q0.3
008        LQ0.3
009        OQ0.0
010        ANI0.0
011        ANI0.1
012        =Q0.0      "Conectar entre común y Q0.3 señal"

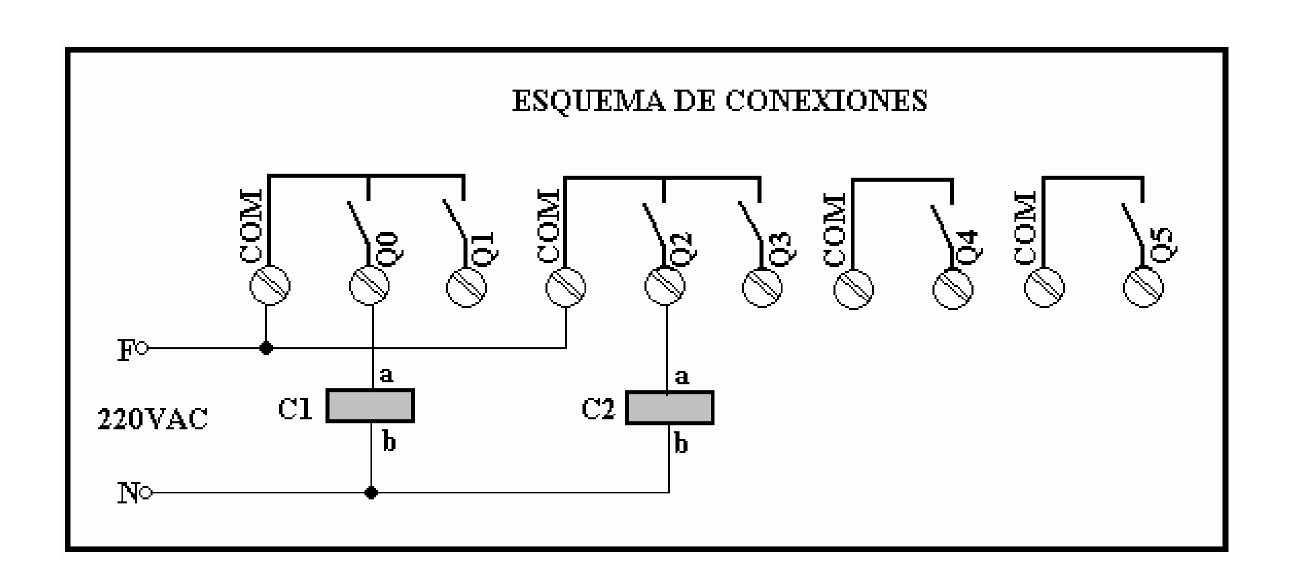
013        TR0

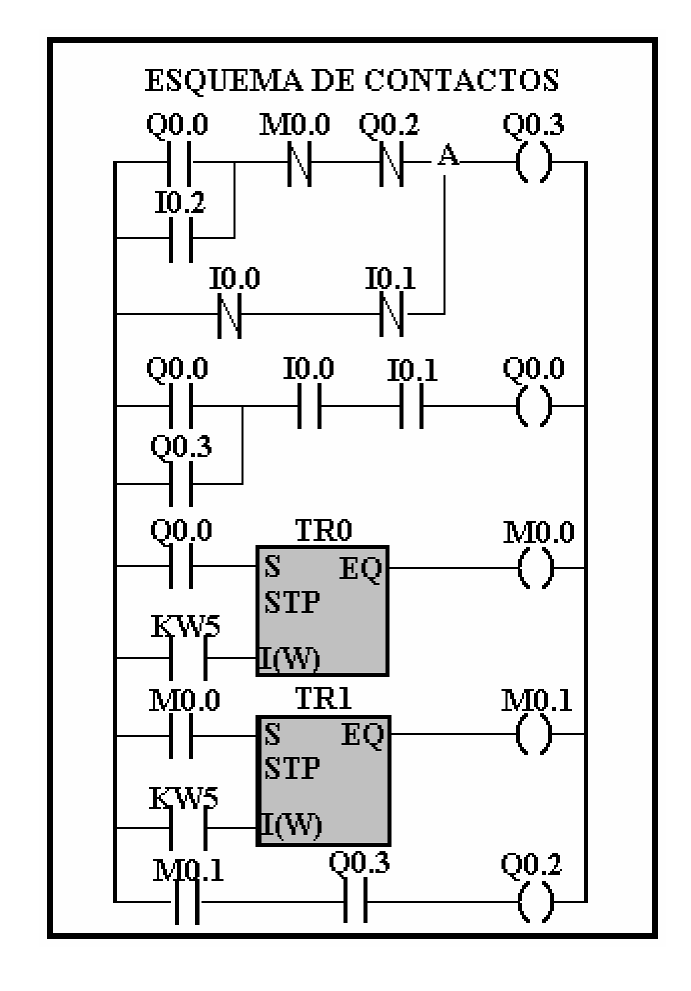
S  :     Q0.0
STP :
IW  :     KW5
EQ :      M0.0

014        TR1

S :      M0.0
STP :
IW :      KW5
EQ :      M0.1

015        LM0.1
016        ANQ0.3
017        =Q0.2
018        EP
```


ESQUEMA DE CONEXIONES
COM
Q0
Q1
COM
Q2
Q3
COM
Q4
COM
Q5
F
220VAC
C1
a
b
C2
a
b
N


ESQUEMA DE CONTACTOS
Q0.0
M0.0
Q0.2
A
Q0.3
I0.2
I0.0
I0.1
Q0.0
I0.0
I0.1
Q0.0
Q0.3
Q0.0
TR0
S
EQ
STP
I(W)
M0.0
KW5
M0.0
TR1
S
EQ
STP
I(W)
M0.1
KW5
M0.1
Q0.3
Q0.2

2.-(CINTA) COMANDO CON INVERSIÓN AUTOMÁTICA TEMPORIZADA:

LISTA DE INSTRUCCIONES

```
0000 BLOQUE 0    "INVERAUT"

000          TR0
               S :          NM0.0
               STP :        I0.1
               IW  :        KW100
               EQ :         M0.1

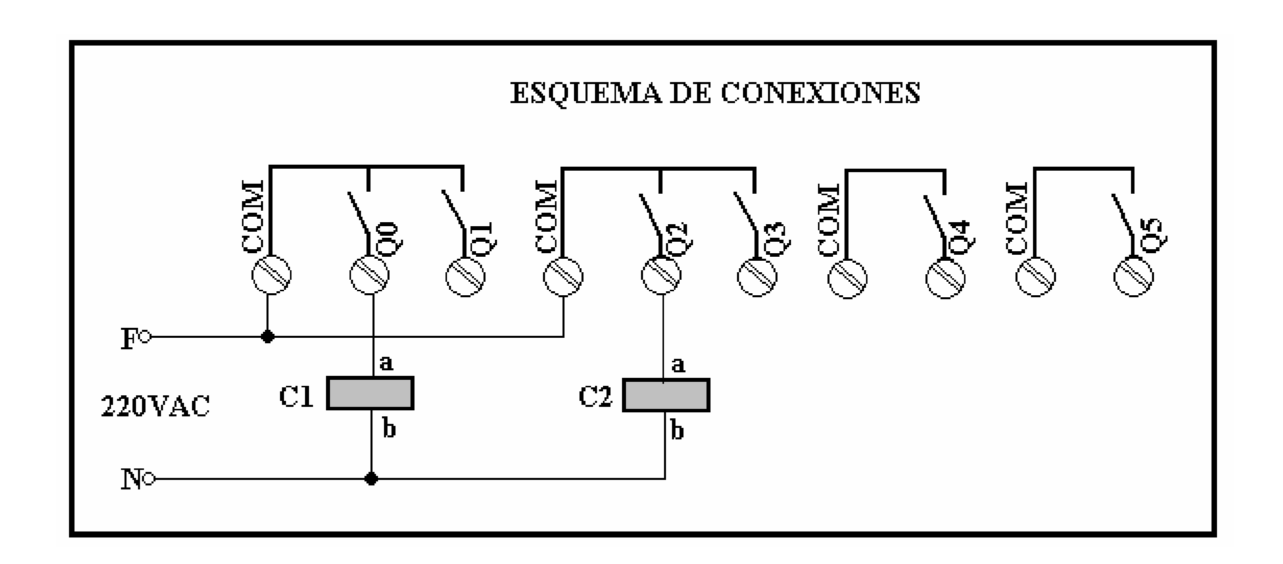
001          TR1
               S :          M0.1
               STP :        I0.1
               IW :         KW100
               EQ :         M0.0

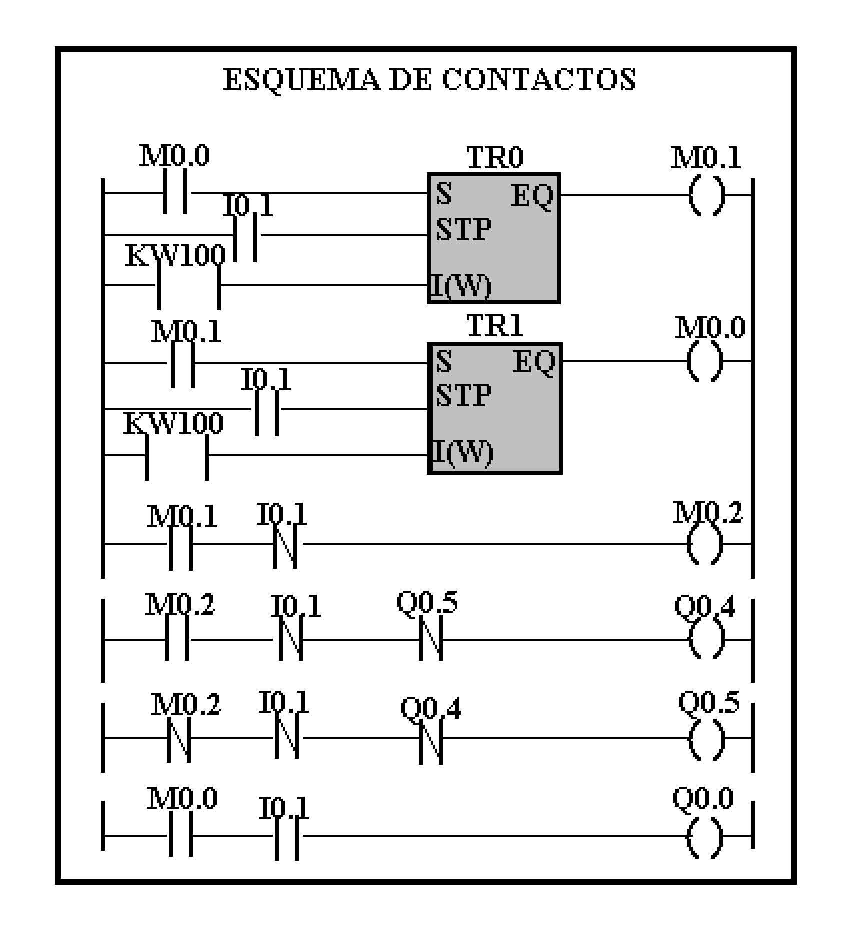
0001 BLOQUE 1    "
002            LM0.1
003            ANI0.1
004            =M0.2

0002 BLOQUE 2
005            LM0.2
006            ANI0.1
007            ANQ0.5
008            =Q0.4

0003 BLOQUE 3
009            LM0.2
010            ANI0.1
011            ANQ0.4
012            =Q0.5

0004 BLOQUE 4
013          LNM0.0
014          ANI0.1
015          =Q0.0
016          EP
```


ESQUEMA DE CONEXIONES
COM
Q0
Q1
COM
Q2
Q3
COM
Q4
COM
Q5
F
220VAC
C1
a
b
C2
a
b
N


ESQUEMA DE CONTACTOS
M0.0
I0.1
KW100
TR0
S
EQ
STP
I(W)
M0.1
M0.1
I0.1
KW100
TR1
S
EQ
STP
I(W)
M0.0
M0.1
I0.1
M0.2
M0.2
I0.1
Q0.5
Q0.4
M0.2
I0.1
Q0.4
Q0.5
M0.0
I0.1
Q0.0

CINTA 1 (INVERSOR DE MARCHA PARA MOTOR MONOFASICO CON CONDENSADOR PERMANENTE

LISTA DE INSTRUCCIONES

0000 BLOQUE 0
001 LI0.0
002 Q0.4
003 OM0.3
004 ANQ0.5
005 ANM0.0
006 =Q0.4

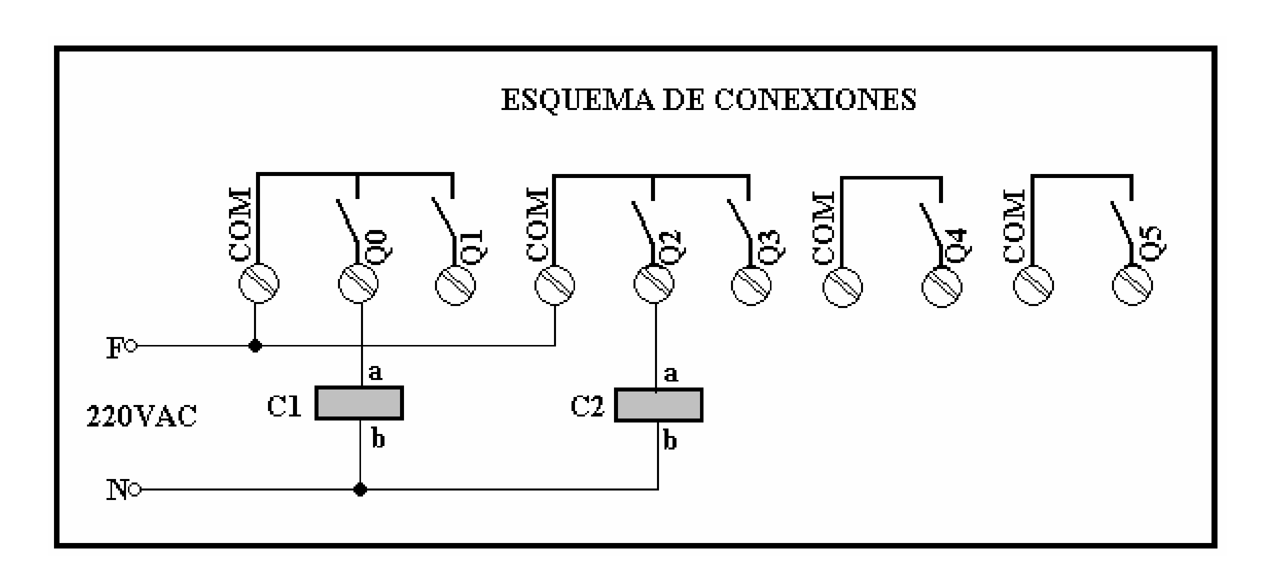
0001 BLOQUE 1
007 TR0
S : Q0.4
STP : I0.1 "Pausa para marcha o detención permanente"
IW : KW50
EQ : M0.0

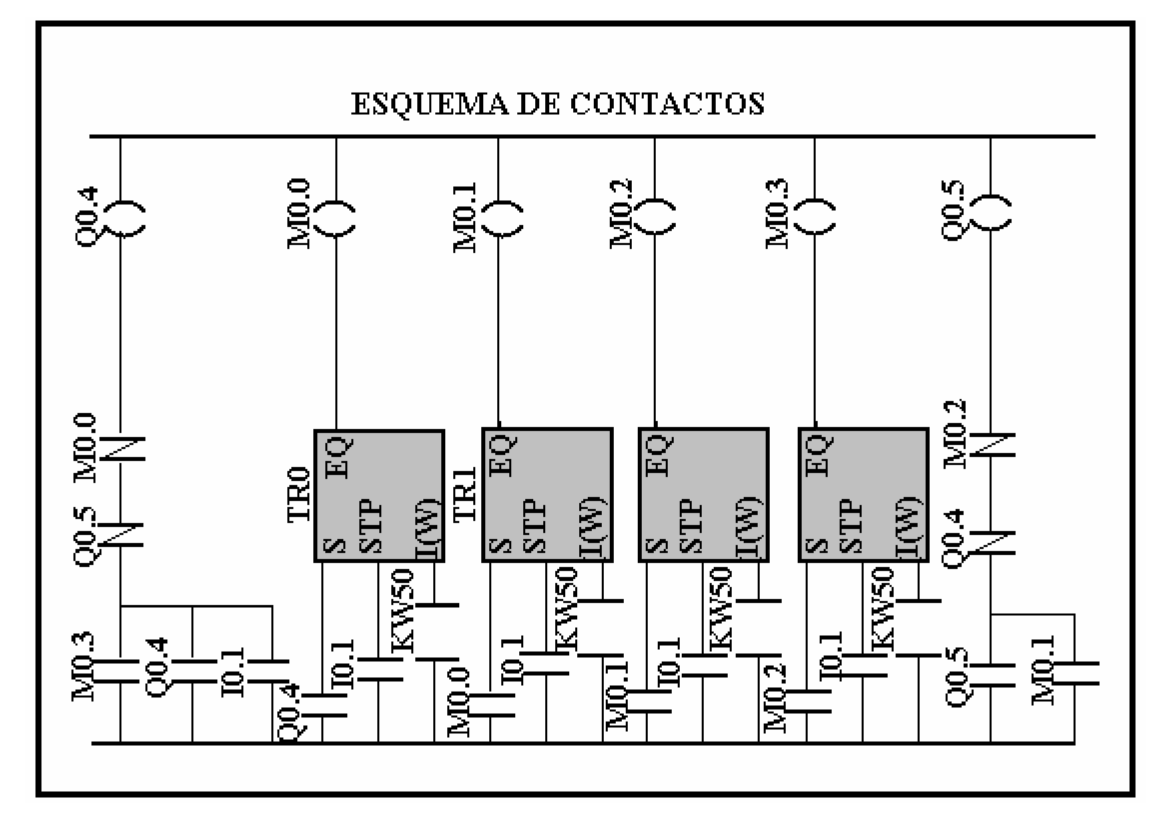
008 TR1
S : NM0.0
STP : I0.1
IW: KW50
EQ : M0.1

009 TR2
S : M0.1
STP : I0.1
IW : KW50
EQ : M0.2

010 TR3
S : M0.2
STP : I0.1
IW : KW50
EQ : M0.3

0002 BLOQUE 2
011 LM0.1
012 OQ0.5
013 ANQ0.4
014 ANM0.2
015 =Q0.5
16 EP

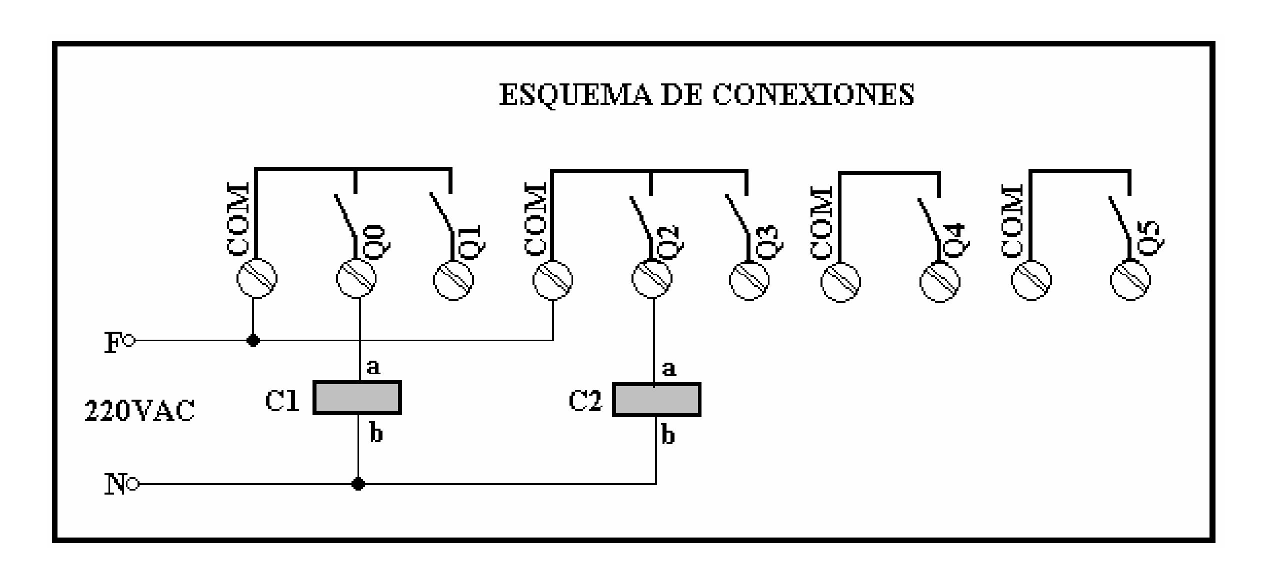

ESQUEMA DE CONEXIONES
COM
Q0
Q1
COM
Q2
Q3
COM
Q4
COM
Q5
F
N
220VAC
C1
C2
a
b
a
b

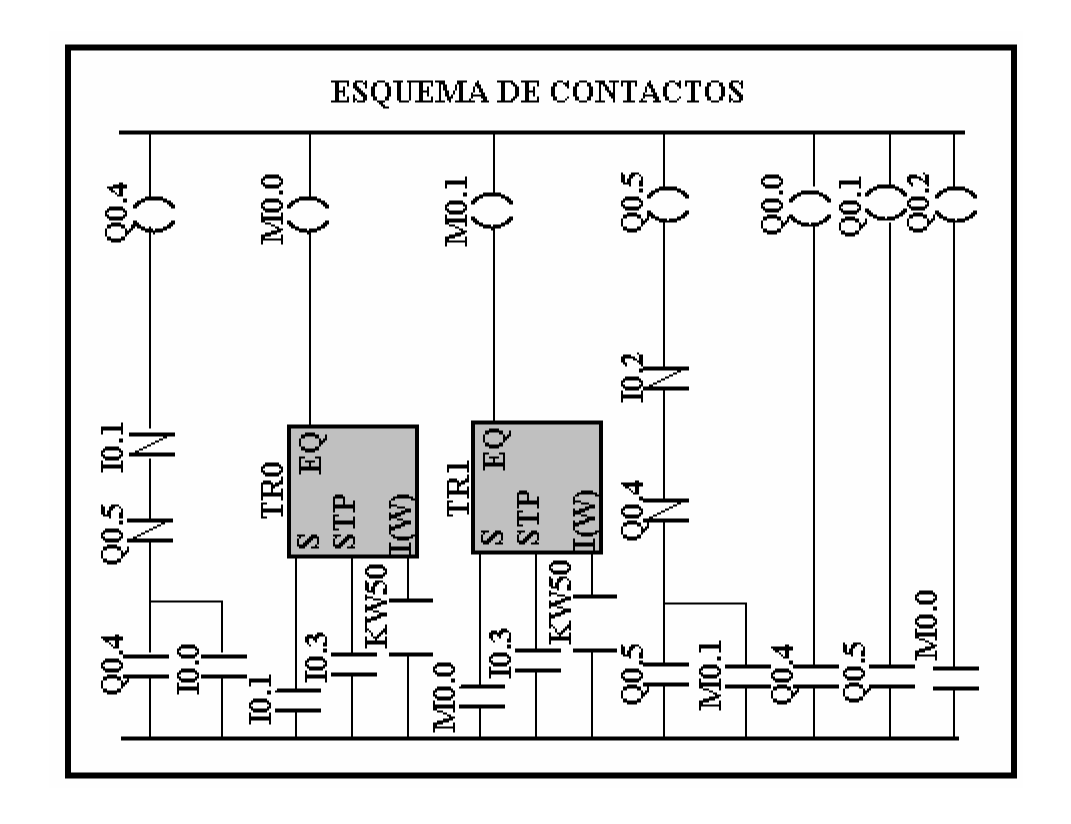

ESQUEMA DE CONTACTOS
Q0.4
M0.0
M0.1
M0.2
M0.3
Q0.5
M0.0
Q0.5
TR0
TR1
EQ
S
STP
I(W)
KW50
M0.2
Q0.4
M0.3
Q0.4
I0.1
Q0.4
I0.1
M0.0
I0.1
M0.1
I0.1
M0.2
I0.1
Q0.5
M0.1

(NADA) PROGRAMA PARA PORTÓN DE CORREDERA AUTOMÁTICO CON PAUSA VARIABLE REGULADA Y LIMITES DE CARRERA

LISTA DE INSTRUCCIONES

0000 BLOQUE 0		"Control abrir"	
001	LI0.0		Start
002	OQ0.4		Enclav.
003	ANQ0.5		Protección
004	ANI0.1		LCI
005	=Q0.4		Bob C1
0001 BLOQUE 1		"Temporización"	
006	TR0		
	S :	I0.1	
	STP :	I0.3	pausa permanente
	IW :	KW100	
	EQ :	M0.0	
007	TR1		
	S :	M0.0	
	STP :	I0.3	Pausa permanente
	IW :	KW100	
	EQ :	M0.1	
0002 BLOQUE 2		"Control cerrar"	
008	LM0.1		
009	OQ0.5		
010	ANQ0.4		
011	ANI0.2	LCD	
012	=Q0.5	Bob C2	
0003 BLOQUE 3		"Señaliz ación"	
013	LQ0.4		
014	=Q0.4		
0004 BLOQUE 4		"Señalización"	
015	LQ0.5		
016	=Q0.1		
0005 BLOQUE 5		"Señalización"	
017	LM0.0		
018	=Q0.2		
019	EP		


ESQUEMA DE CONEXIONES
COM
Q0
Q1
COM
Q2
Q3
COM
Q4
COM
Q5
F
a
C1
220VAC
b
C2
a
b
N

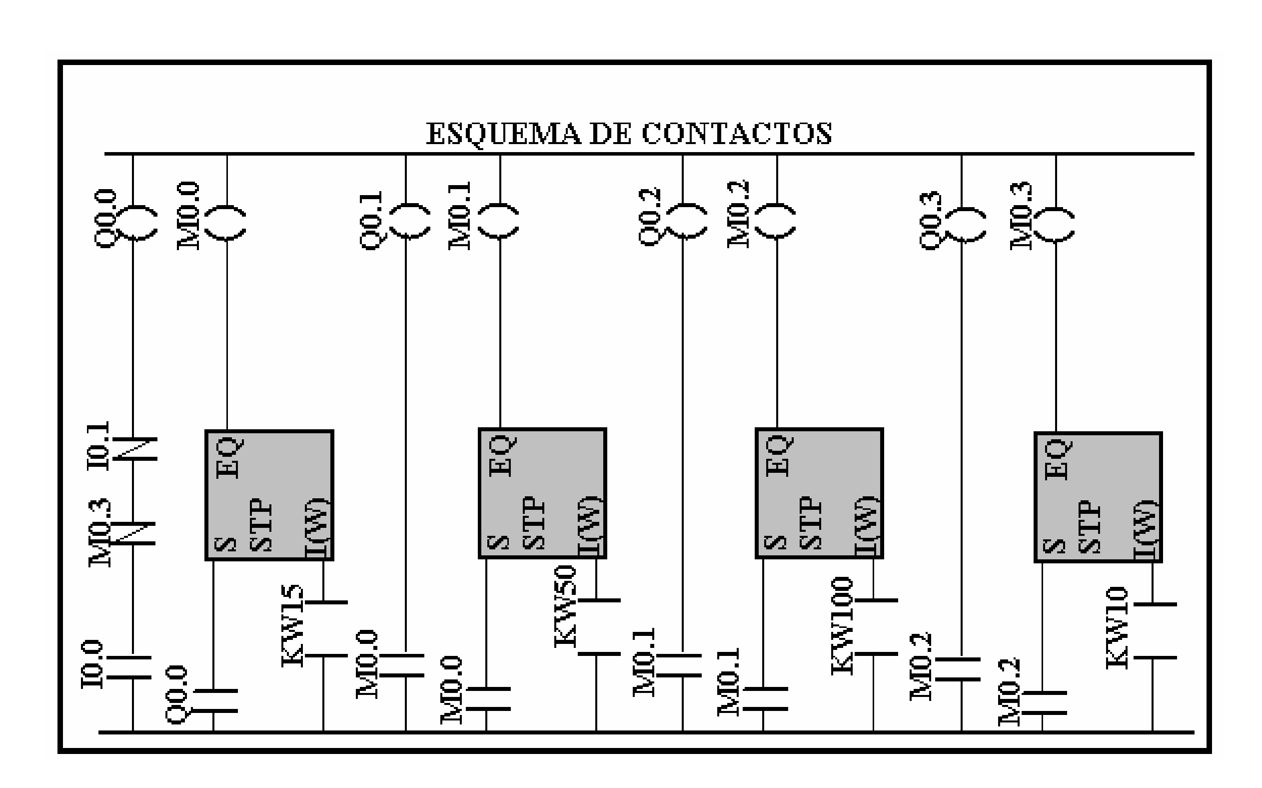

ESQUEMA DE CONTACTOS
Q0.4
M0.0
M0.1
Q0.5
Q0.0
Q0.1
Q0.2
I0.2
I0.1
TR0
EQ
TR1
EQ
Q0.5
S
STP
I(W)
S
STP
I(W)
Q0.4
KW50
KW50
Q0.4
I0.0
I0.3
I0.3
Q0.5
M0.1
Q0.4
Q0.5
M0.0
I0.1
M0.0

(TT2) ACTIVACIÓN SECUENCIAL POR PASOS CON REACTIVACIÓN AUTOMÁTICA

LISTA DE INSTRUCCIONES

Nota: Mantener presionada Start.

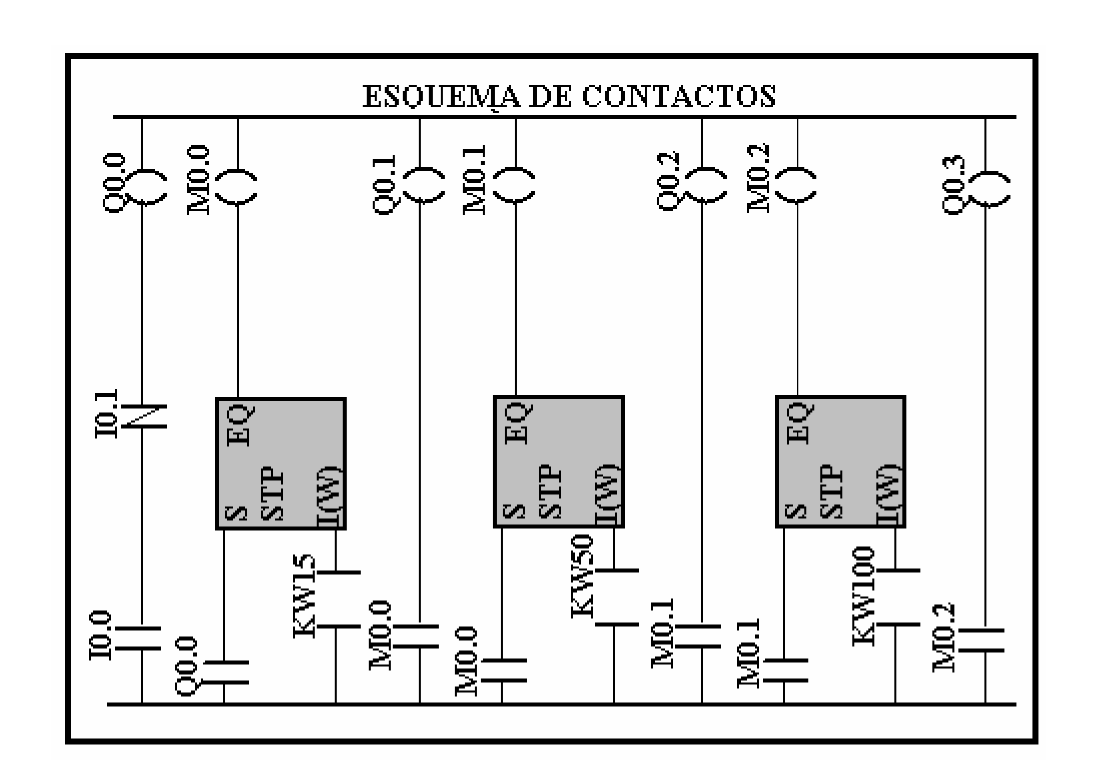
```
0000 BLOQUE 0
001        LI0.0                Start
002        ANM0.3
003        ANI0.1               Stop
004        =Q0.0
005          TR0
              S:         Q0.0
             STP  :
             IW:          KW15
             EQ  :        M0.0
006          LM0.0
007          =Q0.1
008          TR1
              S  :        M0.0
             STP  :
             IW  :        KW50
             EQ  :        M0.1
009          LM0.1
010          =Q0.2
011          TR2
              S  :        M0.1
             STP  :
             IW  :        KW100
             EQ  :        M0.2
             LM0.2
             =Q0.3
012          TR3
              S  :        M0.2
              STP  :
              IW  :       KW10
              EQ  :       M0.3
              EP
```


ESQUEMA DE CONTACTOS
Q0.0
M0.0
Q0.1
M0.1
Q0.2
M0.2
Q0.3
M0.3
I0.1
M0.3
I0.0
Q0.0
S
STP
EQ
I(W)
KW15
M0.0
M0.0
KW50
M0.1
M0.1
KW100
M0.2
M0.2
KW10

(TT1) ACTIVACIÓN SECUENCIAL POR PASOS SIN REACTIVACIÓN AUTOMÁTICA:

LISTA DE INSTRUCCIONES

```
0000 BLOQUE 0
001        LI0.0                 Start
002        ANI0.1                Stop
003        =Q0.0
004        TR0
             S  :       Q0.0
             STP  :
             IW :       KW15
             EQ  :      M0.0
005          LM0.0
006          =Q0.1
007        TR1
             S  :       M0.0
             STP  :
            IW  :       KW50
            EQ  :       M0.1
008         LM0.1
009         =Q0.2
010        TR2
             S  :       M0.1
            STP  :
            IW  :       KW100
            EQ  :       M0.2
011         LM0.2
012         =Q0.3
013         EP
```

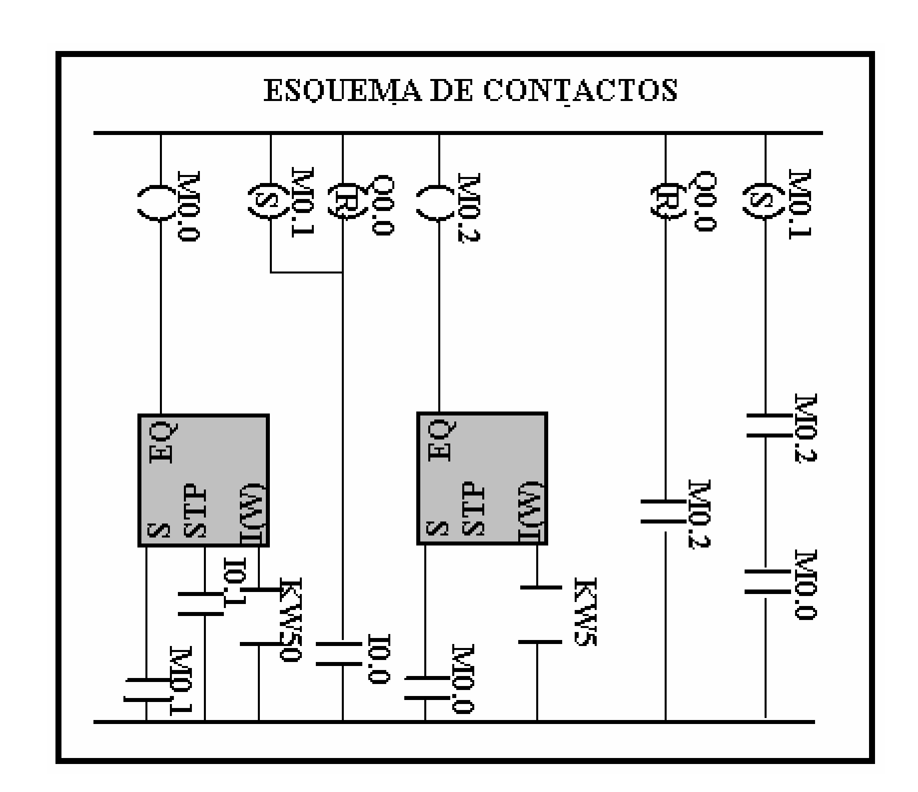

ESQUEMA DE CONTACTOS
Q0.0
M0.0
Q0.1
M0.1
Q0.2
M0.2
Q0.3
I0.1
EQ
STP
S
I(W)
KW15
KW50
KW100
I0.0
Q0.0
M0.0
M0.1
M0.2

TEMPORIZADOR

LISTA DE INSTRUCCIONES

Nota: La salida Q0.0 actúa en forma intermitente al mantenerse accionada I0.0. Con I0.1 se activa stop.

```
0000 BLOQUE 0
001         TR0
              S  :      M0.1
              STP :     I0.1
              IW :      KW50
              EQ :      M0.0
002           LI0.0
003           RQ0.0
004           SM0.1
005           LM0.0
006           SQ0.0
007         TR1
              S  :      M0.0
              STP :
              IW  :     KW5
              EQ  :     M0.2
008           LM0.2
009           RQ0.0
010           LM0.0
011           AM0.2
012           RM0.1
013           EP
```

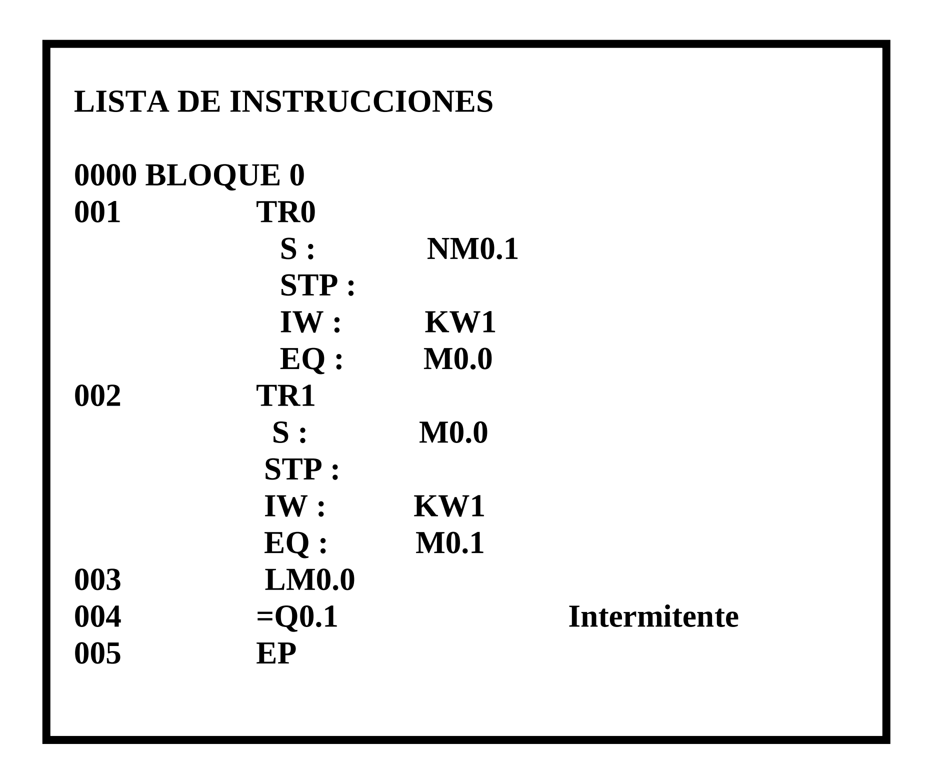

ESQUEMA DE CONTACTOS
M0.0
M0.1
(S)
Q0.0
(R)
M0.2
Q0.0
(R)
M0.1
(S)
EQ
STP
S
T(W)
I0.1
KW50
I0.0
M0.1
EQ
STP
S
T(W)
KW5
M0.0
M0.2
M0.2
M0.0

(TT) ACTIVACIÓN SECUENCIAL POR PASO CON REACTIVACIÓN AUTOMÁTICA:

LISTA DE INSTRUCCIONES

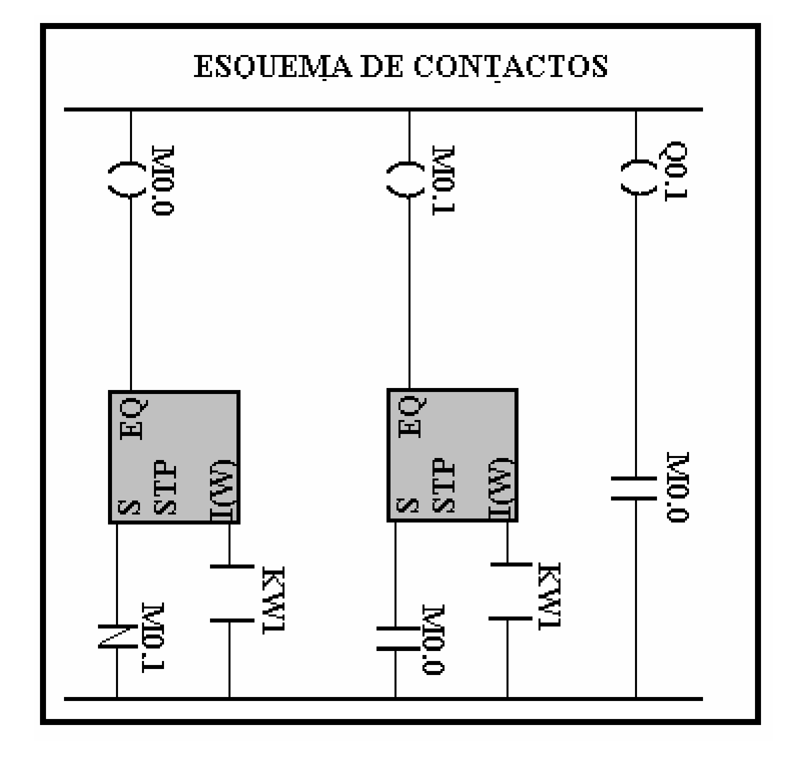
```
0000 BLOQUE 0
001         LI0.0
002         ANM0.3
003         ANI0.1
004         =Q0.0
005         TR0
              S  :        Q0.0
             STP  :
             IW  :        KW15
             EQ  :        M0.0
006          LM0.0
007          ANM0.1
008          =Q0.1
009         TR1
              S :         M0.0
              STP  :
              IW :        KW50
              EQ  :       M0.1
010           LM0.1
011           ANM0.2
012           =Q0.2
013         TR2
              S  :        M0.1
              STP  :
              IW  :       KW100
              EQ  :       M0.2
014           LM0.2
015           ANM0.3
016           =Q0.3
017          TR3
               S  :       M0.2
              STP  :
              IW :        KW10
              EQ  :       M0.3
 018          EP
```

TEMPORIZADOR CICLICO



LISTA DE INSTRUCCIONES

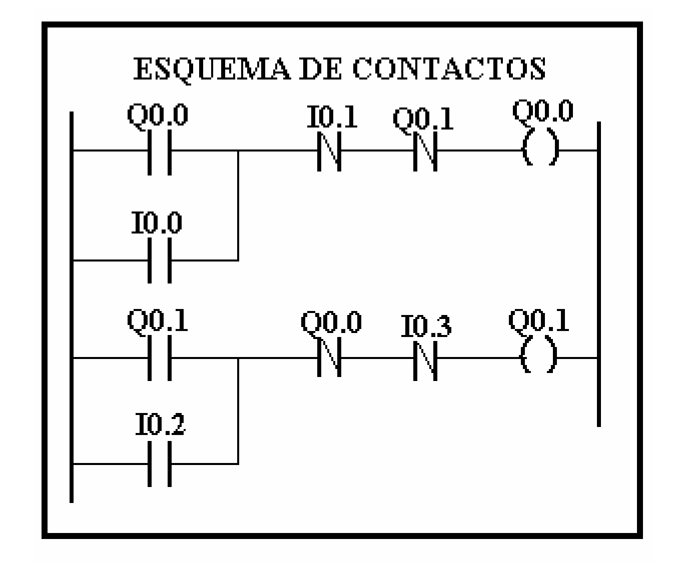
0000 BLOQUE 0			
001	**TR0**		
	S :	**NM0.1**	
	STP :		
	IW :	**KW1**	
	EQ :	**M0.0**	
002	**TR1**		
	S :	**M0.0**	
	STP :		
	IW :	**KW1**	
	EQ :	**M0.1**	
003	**LM0.0**		
004	**=Q0.1**		**Intermitente**
005	**EP**		



(INVERSOR) INVERSOR DE MARCHA PARA MOTOR TRIFÁSICO

LISTA DE INSTRUCCIONES

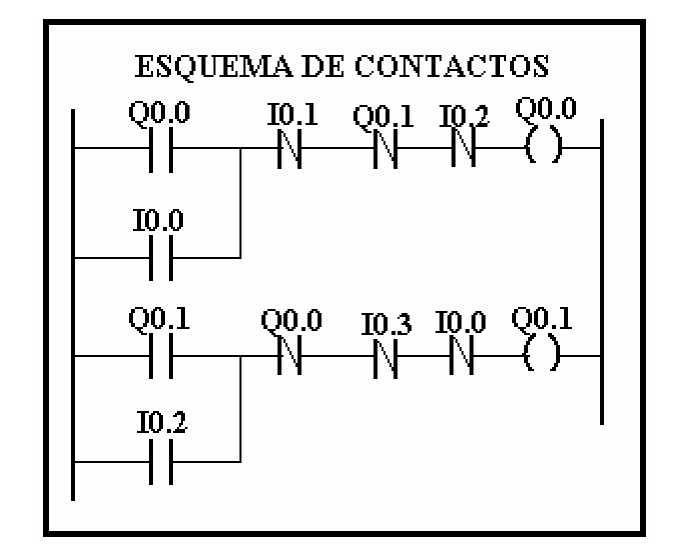
0000 BLOQUE 0		
001	**LI0.0**	**Start Left**
002	**OQ0.0**	**Enclavamiento**
003	**ANI0.1**	**Stop**
004	**ANQ0.1**	**Protección**
005	**=Q0.0**	**Bob Left**
006	**LI0.2**	**Start Right**
007	**OQ0.1**	**Enclavamiento**
008	**ANQ0.0**	**Protección**
009	**ANI0.3**	**Stop**
010	**=Q0.1**	
011	**EP**	



(INVER PRO) INVERSOR DE MARCHA PARA MOTOR TRIFÁSICO CON SISTEMA DE PROTECCIÓN CONTRA CONEXIONES SIMULTÁNEAS:

LISTA DE INSTRUCCIONES

0000 BLOQUE 0		
001	**LI0.0**	**Start left**
002	**OQ0.0**	**Retención**
003	**ANI0.1**	**Stop left**
004	**ANQ0.1**	**Enclavamiento**
005	**ANI0.2**	**Enclavamiento**
006	**=Q0.0**	**Bobina left**
007	**LI0.2**	**Start right**
008	**OQ0.1**	**Retención**
009	**ANQ0.0**	**Enclavamiento**
010	**ANI0.3**	**Stop**
011	**ANI0.0**	**Enclavamiento**
012	**=Q0.1**	**Bobina right**
013	**EP**	



(INVERAUT) INVERSOR CON PROTECCIÓN E INVERSIÓN AUTOMÁTICA:

LISTA DE INSTRUCCIONES

0000 BLOQUE 0

001	LI0.0	Start left
002	OQ0.0	Retención
003	OI0.3	LC right
004	ANI0.4	Enclavamiento
005	ANI0.1	Stop
006	ANQ0.1	Enclavamiento
007	ANI0.2	LC left
008	=Q0.0	Bobina left

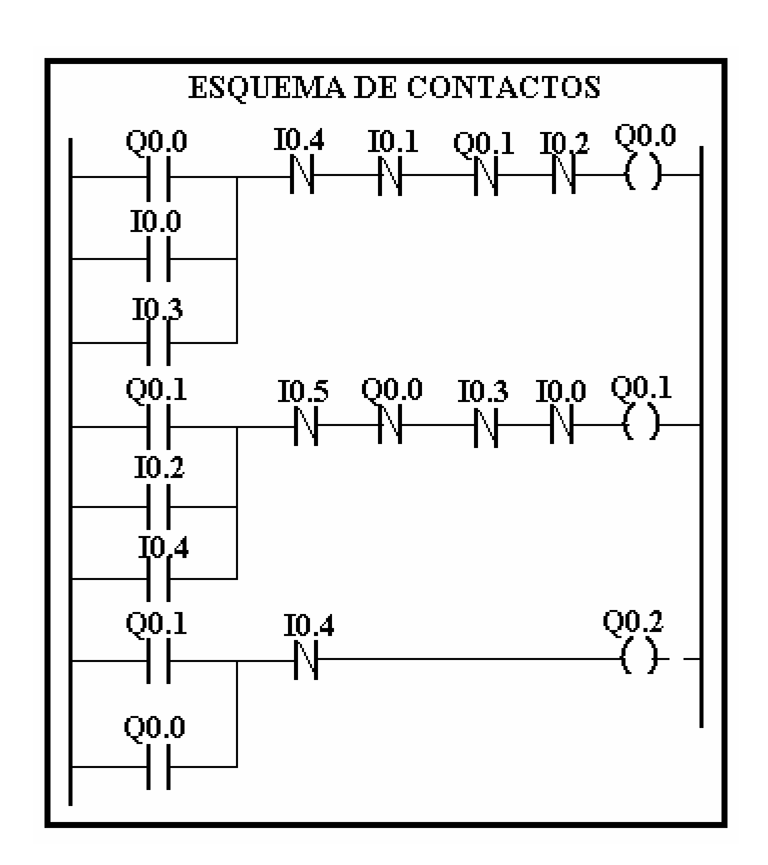
0001 BLOQUE 1

009	LI0.4	Start right
010	OQ0.1	Retención
011	OI0.2	LC left
012	ANQ0.0	Enclavamiento
013	ANI0.5	Stop
014	ANI0.3	LC right
015	ANI0.0	Enclavamiento
016	=Q0.1	Bobina right

0002 BLOQUE 2

017	LQ0.0
018	OQ0.1
019	ANI0.4
020	=Q0.2
021	EP


INSTITUTO
iade
CENTRO DE CAPACITACIÓN


ESQUEMA DE CONTACTOS
Q0.0
I0.4
I0.1
Q0.1
I0.2
Q0.0
I0.0
I0.3
Q0.1
I0.5
Q0.0
I0.3
I0.0
Q0.1
I0.2
I0.4
Q0.1
I0.4
Q0.2
Q0.0





(INVERXO) GENERADOR DE IMPULSOS CADA 10 SEGUNDOS:

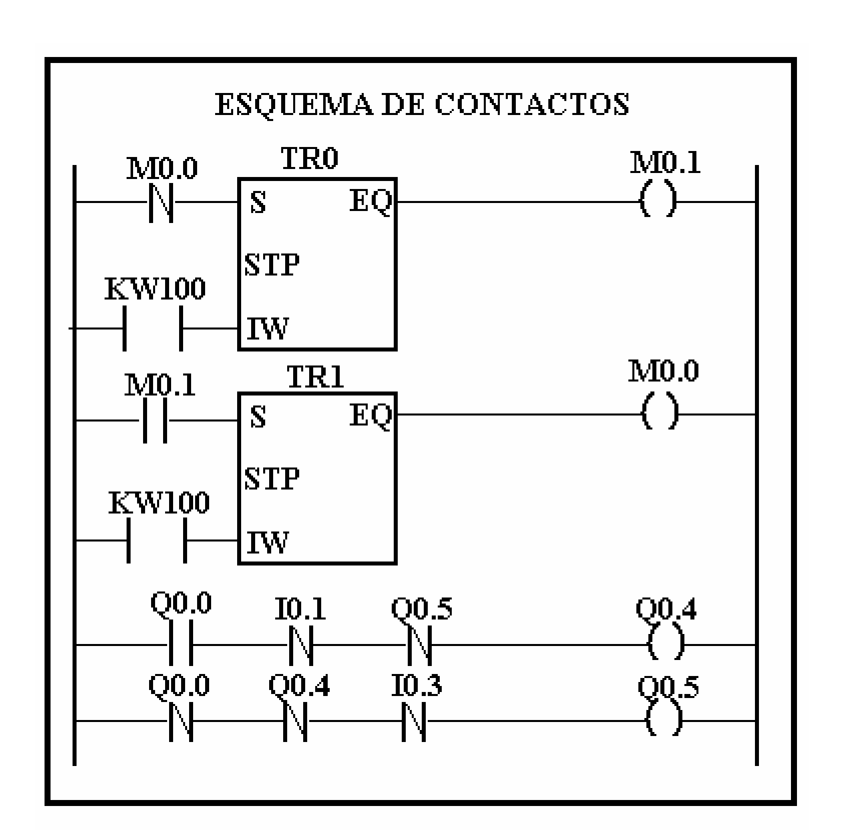
LISTA DE INSTRUCCIONES

0000 BLOQUE 0

001	**TR0**	
	S :	**NM0.0**
	STP :	
	IW :	**KW100**
	EQ :	**M0.1**
002	**TR1**	
	S :	**M0.1**
	STP :	
	IW :	**KW100**
	EQ :	**M0.0**

0001 BLOQUE 1

003	**LM0.1**
004	**ANI0.1**
005	**ANQ0.5**
006	**=Q0.4**
007	**LNM0.1**
008	**ANQ0.4**
009	**ANI0.3**
010	**=Q0.5**
011	**EP**


ESQUEMA DE CONTACTOS
M0.0
TR0
M0.1
S
EQ
STP
KW100
IW
M0.1
TR1
M0.0
S
EQ
STP
KW100
IW
Q0.0
I0.1
Q0.5
Q0.4
Q0.0
Q0.4
I0.3
Q0.5

(ACTUADOR) ARRANCADOR ESTRELLA TRIÁNGULO:

LISTA DE INSTRUCCIONES

0000 BLOQUE 0 "Power"

```
001    LI0.0
002    OQ0.0
003    ANI0.1
004    =Q0.0
```

0001 BLOQUE 1 "Temporizador"

```
005    LIA0.0
006    =MB15.0
007    LKB0
008    =MB15.8

009    TR0

        S   :   Q0.0
        STP :
        IW  :   MW15.0
        EQ  :   M0.0
```

0002 BLOQUE 2 "Estrella"

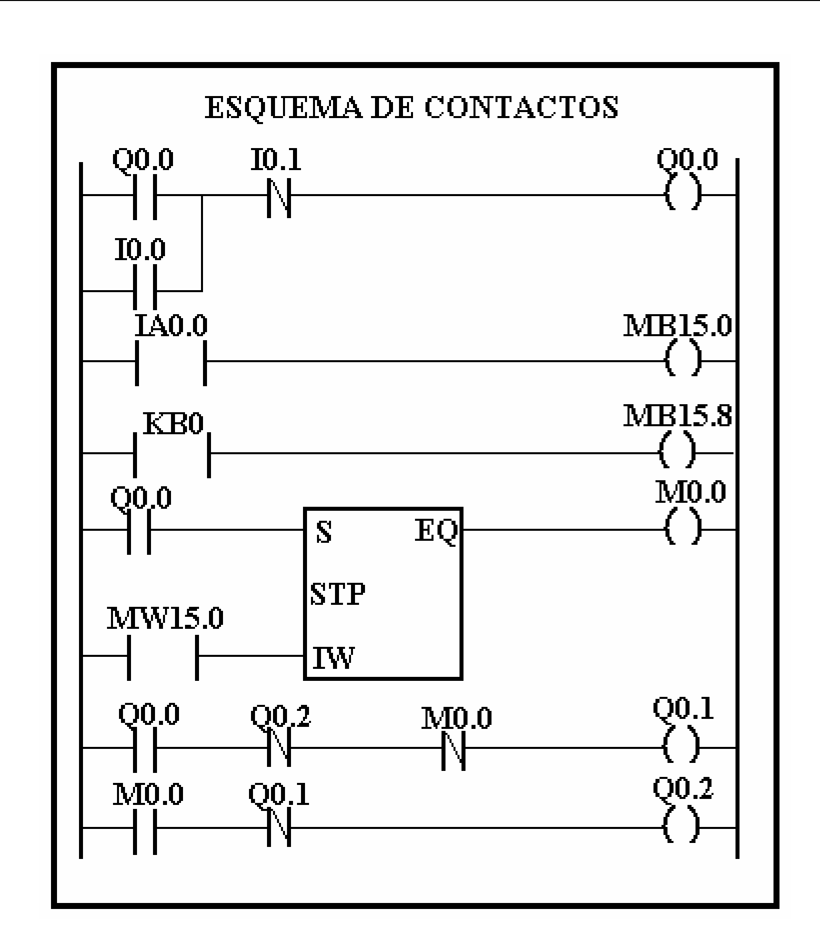
```
010    LQ0.0
011    ANQ0.2
012    ANM0.0
013    =Q0.1
```

0003 BLOQUE 3 "Triángulo"

```
014    LM0.0
015    ANQ0.1
016    =Q0.2
```

0004 BLOQUE 4 " Final"

```
017    EP
```


ESQUEMA DE CONTACTOS
Q0.0
I0.1
Q0.0
I0.0
IA0.0
MB15.0
KB0
MB15.8
Q0.0
M0.0
S
EQ
STP
MW15.0
IW
Q0.0
Q0.2
M0.0
Q0.1
M0.0
Q0.1
Q0.2

(ARRANCADOR) ARRANCADOR ESTRELLA TRIANGULO:

LISTA DE INSTRUCCIONES

0000 BLOQUE 0

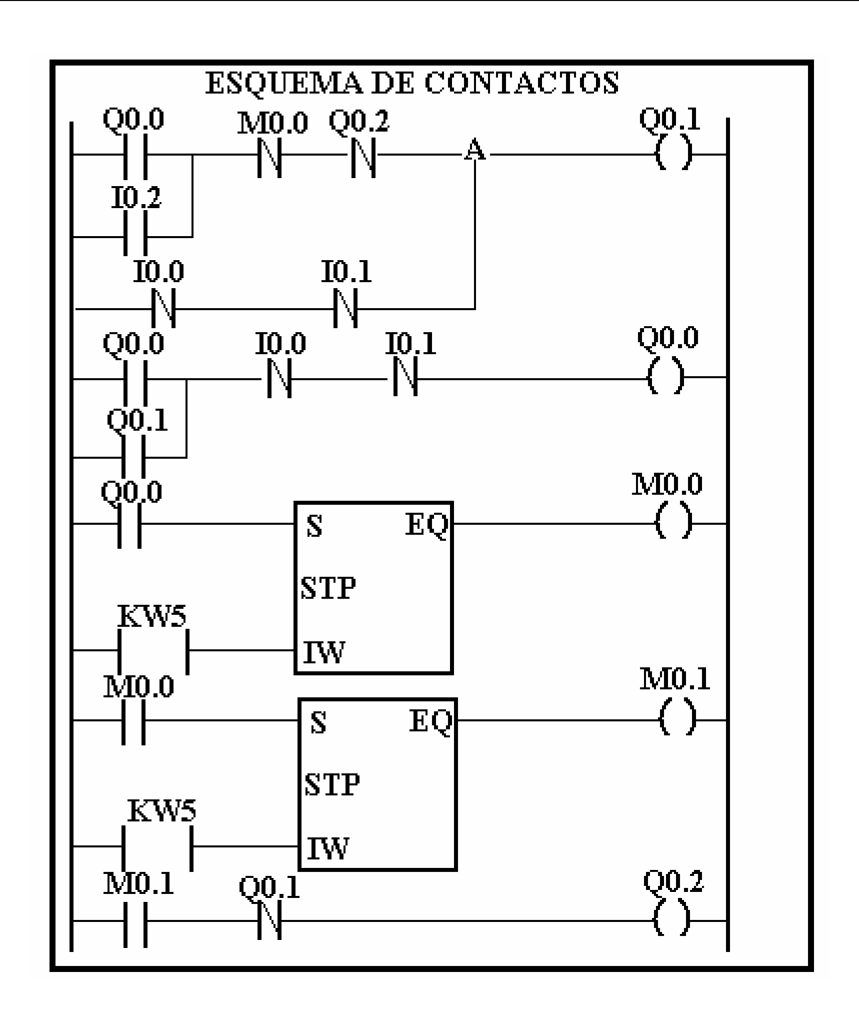
001	**LNI0.0**
002	**ANI0.1**
003	**LI0.2**
004	**OQ0.0**
005	**ANM0.0**
006	**ANQ0.2**
007	**A**
008	**=Q0.1**

0001 BLOQUE 1

009	**LQ0.1**
010	**OQ0.0**
011	**ANI0.0**
012	**ANI0.1**
013	**=Q0.0**

0002 BLOQUE 2

014	**TR0**	
	S :	**Q0.0**
	STP :	
	IW :	**KW80**
	EQ :	**M0.0**
015	**TR1**	
	S :	**M0.0**
	STP :	
	IW :	**KW10**
	EQ :	**M0.1**
016	**LM0.1**	
017	**ANQ0.1**	
018	**=Q0.2**	
019	**EP**	


ESQUEMA DE CONTACTOS
Q0.0
M0.0
Q0.2
A
Q0.1
I0.2
I0.0
I0.1
Q0.0
I0.0
I0.1
Q0.0
Q0.1
Q0.0
S
EQ
M0.0
STP
KW5
IW
M0.0
S
EQ
M0.1
STP
KW5
IW
M0.1
Q0.1
Q0.2

(TEMPO 1)

Nota: LI 0.0 se debe mantener presionada.
La salida Q0.1 se activa después que se completa el tiempo de todos los temporizadores.

LISTA DE INSTRUCCIONES

0000 BLOQUE 0

```
001        LI0.0
002        ANI0.1
003        =Q0.0

004        TR0

             S  :        Q0.0
             STP :
             IW :        KW15
             EQ  :       M0.0

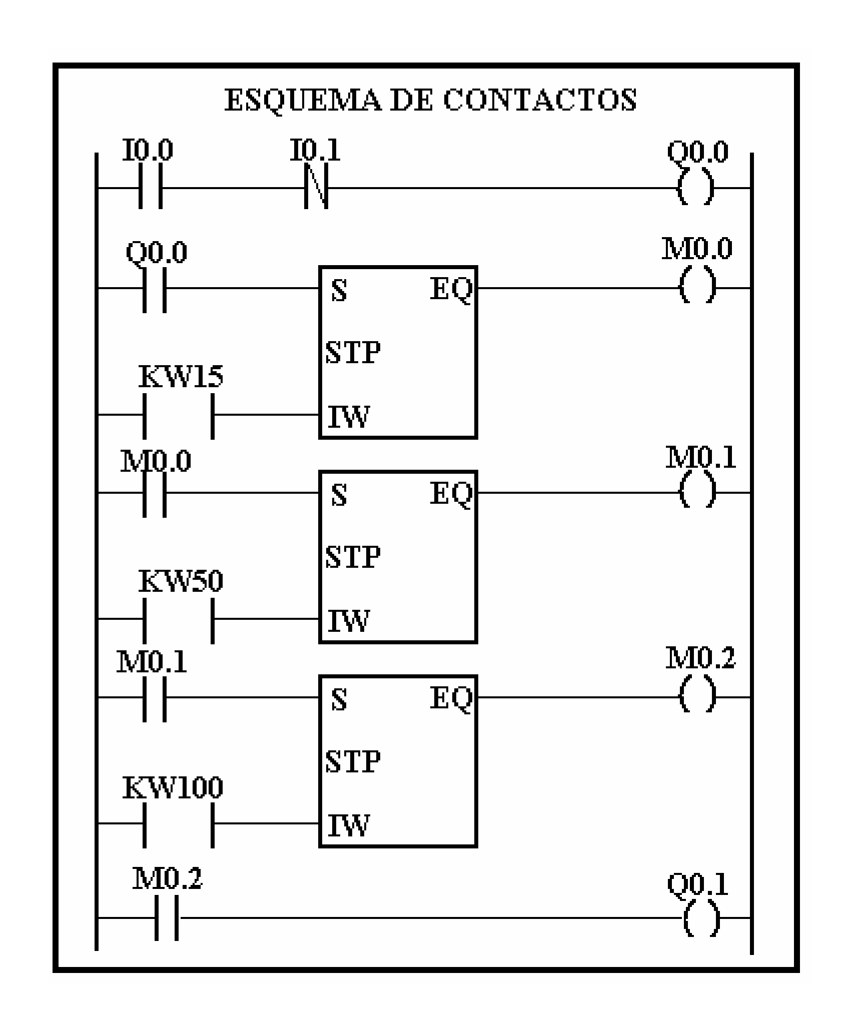
005        TR1

             S  :        M0.0
             STP  :
             IW  :       KW50
             EQ  :       M0.1

006        TR6

             S  :        M0.1
           STP  :
           IW :          KW100
           EQ  :         M0.2

007        LM0.2
008        =Q0.1
009        EP
```


ESQUEMA DE CONTACTOS
I0.0
I0.1
Q0.0
Q0.0
M0.0
S
EQ
STP
KW15
IW
M0.0
M0.1
S
EQ
STP
KW50
IW
M0.1
M0.2
S
EQ
STP
KW100
IW
M0.2
Q0.1

(PRUEBA) PARTIDA Y DESCONEXIÓN A 10 SEGUNDOS:

LISTA DE INSTRUCCIONES

0000 BLOQUE 0 **"Motor 1"**

```
001    LI0.0
002    OQ0.0
003    ANI0.2
004    ANM0.0
005    =Q0.0
```

0001 BLOQUE 1 **"Temporizador"**

```
006    LIA0.0
007    =MB15.0
008    LKB0
009    =MB15.8

010    TR0

        S  :     Q0.0
        STP :
        IW  :    MW15.0
        EQ  :    M0.0
```

0002 BLOQUE 2 **"Motor 2"**

```
011    LNQ0.0
012    =Q0.2
```

0003 BLOQUE 3 **"Final"**

```
013    EP
```

REGISTROS DE DESPLAZAMIENTO
(Shift Register = SR)

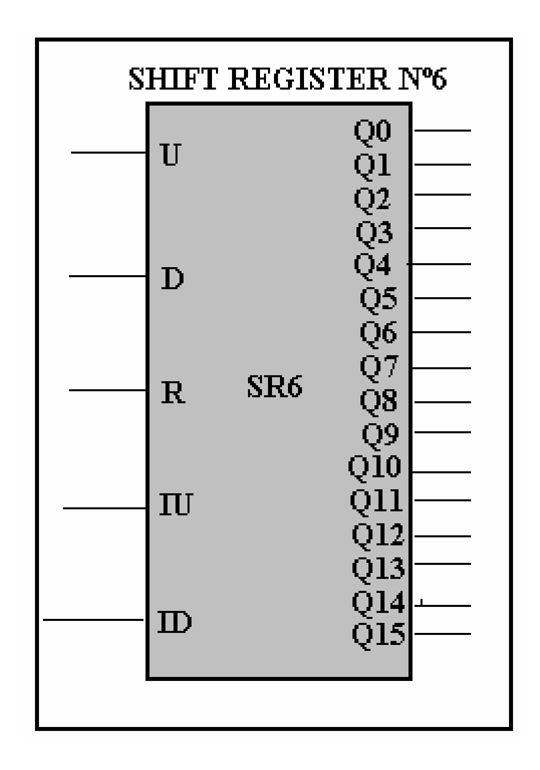
Este es un registro de desplazamiento de1 bit, con una longitud fija de registro.

Cuando hay un flanco ascendente en la entrada **"U"**, el valor de la entrada"**IU"**se acepta el primer campo de registro, después de que todos los demás campos de registros se han desplazado un paso en sentido ascendente. Con un flanco ascendente en la entrada **"D"** el valor **"ID"** se acepta en el último campo de registro, después de que todos los demás campos se han desplazado un paso en sentido descendente. Los contenidos de todos los campos de registro se visualizan a través de las salidas **"Q".**

Cuando la entrada **"R"** está en 1(nivel alto), el **SR** se pone a cero y todos los campos de registro borran. La longitud del registro está limitada a 16 campos de registro.

El PS4 - 111 incluye la posibilidad de programar hasta 32 SR.

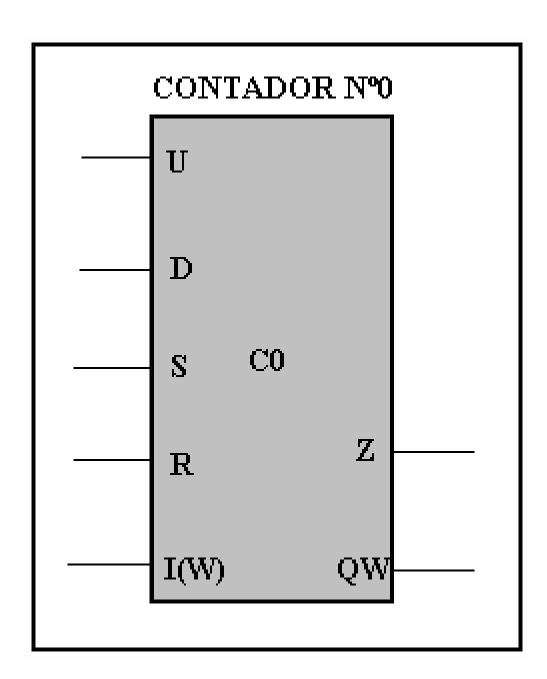
El diagrama en bloques del SR es:



CONTADORES (C)

Incluimos esta instrucción dentro de este grupo, por ser, también, muy utilizada en los esquemas de contactos o listas de instrucciones. El **PS4 - 111** incluye la posibilidad de programar hasta 32 contadores, con un margen de contaje de 0 a 65535. Cuando hay un flanco ascendente en la entrada **"S",** el valor presente en **"I"** es aceptado en el contador. El contaje incremental tiene lugar cuando hay un flanco ascendente en la entrada **"U"** y el contaje decremental cuando hay un flanco ascendente en la entrada **"D".** Cuando la entrada **"R"** está en alto, el contador se pone a cero y el contenido se borra. La salida **"Z"** del contador es igual a 1, cuando el contenido del contador es igual a cero. La salida **"Q"** indica la lectura del contador en uso.

Diagrama en bloques del contador



FICHA PRÁCTICA DE CONTADORES

El diagrama en bloques de un contador es:
En nuestro autómata podemos disponer de hasta 32 contadores, los cuales se llaman a través de la letra C (desde C0 hasta C31).

Explicación de las entradas y salidas

Entradas:

U (Up) ---------------------------------------Impulso incremental....................(Bit)
D (Down)-----------------------------------Impulso decremental...................(Bit)
S (Set)--------------------------------------- Puesta a 1....................................(Bit)
R (Reset)------------------------------------Puesta a cero...............................(Bit)
I (Input)------ --------------------------Valor de entrada de puesta a 1....(Word)

Salidas:

Z---Lectura contador cero....................(Bit)
Q--Lectura contador............................(Word)

Descripción:

Existiendo un flanco ascendente en la entrada **"S",** el valor presente en **"I"** es aceptado en el contador.
El contaje incremental tiene lugar cuando existe un flanco ascendente en la entrada **"U"** y el contaje decremental cuando hay un flanco ascendente en la entrada **"D".**
Cuando la entrada **"R"** esta en alto, el contador se pone a cero y el contenido se borra. La salida **"Z"** del contador es igual a 1 cuando el contenido del contador es igual a 0.
La salida **"Q"** indica la lectura del contador en uso.

Un ejemplo de programación, para un contaje ascendente, es el siguiente:

a)En lista de instrucciones:

C11
U : I0.5 Contaje incremental.
D :
S :
R : I0.6 Reset o puesta a cero.
I :
Z :
Q : QW0 Valor real

Funcionamiento: El contador **C11** ha de contar un paso incremental cada vez que **I0.5** cierra. El contador se pone a 0 con **I0.6**. La lectura real del contador se visualiza a través de la salida **QW0.**
El contaje decremental y la puesta a 1 no se utilizan.

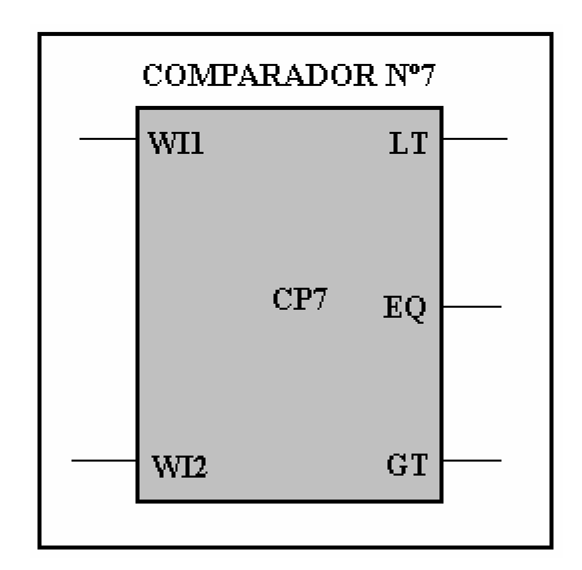
COMPARADORES (CP)

El módulo compara los valores en las entradas de palabra I1 e I2 y a continuación pone las salidas conforme a la tabla de trabajo.

Tabla de trabajo:

I1< I2	**GT = 0**	**EQ = 0**	**LT = 1**
I1 = I2	**GT = 0**	**EQ = 1**	**LT = 0**
I1 > I2	**GT = 1**	**EQ = 0**	**LT = 0**

El diagrama en bloques del comparador es:



SOFTWARE DE PROGRAMACION

Después de la instalación del software de programación y de seleccionar el **SUCOsoft S30-S3** entrando **SUCOS3**, aparecerá en pantalla el mensaje de que el **SUCOsoft S 30-S3** ya está cargado en la memoria del **PC**.
En este display aparece entre otras cosas el Nº de versión del paquete de software.

```
KLOCKNER - MOELLER

SUCOSOFT S30
SUCOS PS 3

(c) Copyright by Klockner - Moeller 1986 ...1991

Versión 2. 31 del 7.2 .1994 solo utilizable con CPU versión V1. 0

Seguir con (INTRO)
```

Después de pulsar la tecla **INTRO** se visualiza en la pantalla, el interface serie del PC **COM1** para la conexión del convertidor externo de interface.

```
SUCOSOFT S30

ELECCIÓN DEL INTERFACE
Interface : COM1

Se ha elegido el interface
```

A continuación aparece el **menú principal:**

```
SUCOSOFT S30

MENU PRINCIPAL

F1  PROGRAMACIÓN
F2  TEST/PUESTA EN MARCHA
F3  DOCUMENTACIÓN
F4  TRANSFERENCIA
F5  COMPARACIÓN
F6  ADMINISTRACIÓN DE ARCHIVOS
F7  ELECCIÓN DEL SISTEMA
F8  MS - DOS

F10  AYUDA

Elija por favor

Camino : C:\PS3\U23\AWB
```

Desde el **menú principal** se accede a los diversos **submenús** a través de las distintas **teclas de función (F1) a (F10**). Las entradas y datos necesarios se llevan a cabo de modo interactivo.
Pulsando la tecla de función **(F10) AYUDA** se obtienen aclaraciones y descripciones referentes a cada una de las diferentes funciones.
Después de salir de la **función de ayuda**, el programa regresa automáticamente al menú previamente seleccionado.

```
SUCOSOFT S30

PROGRAMACIÓN

F1   MENU PRINCIPAL
F2   PROGRAMACIÓN LDI
F3   PROGRAMACIÓN EDC
F4   PROGRAMACIÓN EBF
F5   LISTA DE REFERENCIA
F6   COMPILADOR
F10  AYUDA

Elija por favor

Camino : C:\PS3\U23\AWB
```

(F1) MENU PRINCIPAL.

Retorno al menú principal

(F2) PROGRAMACION EN LDI.

Aquí se hallan las funciones con las que se puede entrar y modificar programas y listas de referencias.

(F3) PROGRAMACION EN EDC.

Idem. a (F2).

(F4) PROGRAMACION EN EDF.

Idem. a (F2).

(F5) DATOS DE REFERENCIAS.

Con ésta función se puede crear y ampliar el fichero de referencias.
El fichero de referencias contiene los operandos simbólicos y los asigna a lo s operandos absolutos correspondientes. Este fichero contiene, además, campos en los cuales pueden entrarse datos sobre el tipo de contacto, los números de los terminales y el significado general del operando.

(F6) COMPILADOR.
Antes de la compilación, se verifica si el programa contiene algún error y en caso negativo se traduce al lenguaje del autómata.

(F10) AYUDA.
Selección de textos de ayuda.

ESTABLECIMIENTO DEL NOMBRE DEL PROGRAMA DE USUARIO
(Fichero fuente).

```
SUCOSOFT S30

Especifique un nombre para : Programa fuente

    Nombre : Ejercici

    Disco : c

       Con (F1) hasta (F10) puede interrumpir la introducción del nombre

Camino : C:\PS3\U23\AWB
```

Pulsando la tecla de función (**F2) PROGRAMACION EN LDI**, se visualiza el display anterior. Observar las siguientes instrucciones para entrar el nombre del fichero:

a)Ha de tener como máximo 8 caracteres alfanuméricos, el primero de los cuales ha de ser una letra.

b)Posibilidades de corrección: Borrar carácter por carácter con la tecla **RETROCESO** o nueva entrada del nombre de fichero después de responder a la pregunta: **"Desea repetir la entrada del nombre"** con (S) Si.

c)Completar la entrada del nombre con **INTRO**.
A continuación:

d)Especificación de la unidad de disco con:

a: Para el diskette en la unidad "a", o bien
c: Para el disco duro.

e)Completar la entrada con **INTRO**. Ahora se le preguntará el nombre para el fichero de referencias.

ESTABLECIMIENTO DEL NOMBRE DE L PROGRAMA DE USUARIO
(Fichero de referencias).

```
SUCOSOFT S30

Especifique un nombre para :    Lista de referencia

    Nombre  : Ejercici

    Disco : c

        Con (F1) hasta (F10) puede interrumpir la  introducción  del nombre

Camino : C:\PS3\U23\AWB
```

Aquí rigen las mismas condiciones que para la entrada del nombre del programa de usuario **(Fichero fuente).**

a)Pulse **INTRO** si el **fichero fuente** y el **fichero de referencias** tienen el mismo nombre. El nombre será aceptado.

b)Pulse **INTRO** si el **fichero fuente** y el **fichero de referencias** se han de almacenar en la misma unidad de disco.

c)Es posible asignar nombres diferentes al **fichero fuente** y al **fichero de referencias**.

d)Es posible especificar dife rentes unidades de disco para el **fichero fuente** y el **fichero de referencias**.

EDITOR DE PROGRAMAS

```
MENU PRINCIPAL ---- PROGRAMACIÓN ------------c: ejercici . q3 ------c:  ejercici . z3

F1  RETORNO                F4   SALVAR DATOS       F7    SALIDA  PROGRAMA  FUENTE
F2  EDITAR PROGRAMA                                F8    SALIDA LISTA DE REFERENCIA
F3  EDITAR LISTA REF.                              F9    SALIDA FORMATO PRG 3
```

1 Especificación del tipo de PLC: PS3.

2 Programa que se está procesando:

c: Ejercicio.q(z)3

Fichero de referencias (Operandos, comentarios, terminales de conexión)

Fichero fuente.

Nombre de fichero: Ejercicio.

Especificación de la unidad de disco: C

```
 00000  BLOQUE0   "Inicio
 001               LI0.0                Pulsador  S0  accionado
 002               AI0.1                Pulsador  S1  accionado
 003               OI0.2                Interruptor S2 con
 004               =Q0.0                indicador luminoso H0  con.
 00001  BLOQUE1   "Tiempo
 001               TR0
 002             (   ) S :     I0.3     Arranque  TR0
 003             (   ) STP :   I0.4     Interrupción TR0
 004              (w) I  :  KW50        Tiempo transcurrido  = 5 s.
 005             (   ) EQ : Q0.1        Motor  M1 con.
 00002  BLOQUE2  " Fin
 001                EP

      MENU PRINCIPAL ---- PROGRAMACIÓN——EDITAR————INSERTAR
 F1  RETORNO            F4  AÑADIR LINEA        F7  BORRAR BLOQUES
 F2  ABRIR BLOQUE       F5  BUSCAR /SUSTITUIR   F8  BORRAR LINEA ACTUAL
 F3  BUSCAR BLOQUE      F6  COPIAR BLOQUES      F9  COMENTARIO OPERANDO
```

(F2) ABRIR BLOQUE. Comienzo de la entrada del programa; en pantalla aparece automáticamente el bloque **Nº 00000**.

a)A continuación sigue la entrada de un nombre de bloque (llamado marca o etiqueta) con un máximo de 8 caracteres alfanuméricos, cuyo primer carácter ha de ser una letra.

b)El comentario de bloque comienza con el signo de comillas **(“)** (62 caracteres como máximo). Si no se desea ningún comentario se pulsará la tecla **INTRO**, con lo que se abrirá la primera línea de programa **001**.

c)En la programación es posible utilizar minúsculas y/o mayúsculas.

d)Concluir una instrucción, por ej. **LI0.0** con la tecla **INTRO**. El cursor salta al principio del comentario.

e)Ingresar el comentario. A continuación completar con la tecla **INTRO**. El cursor salta a la siguiente línea de programa.

f)Las entradas de los programas terminan siempre con la instrucción **EP (End of Program.).**

g)Después de pulsar **INTRO**, una vez concluida la entrada del comentario de bloque se abrirá la línea número **002.**
Continúe con la programación del modo habitual.

h)Los números de los bloques que siguen al bloque aumentan automáticamente en 1.

(F3) SELECCIONAR BLOQUE.

a)Llamada de un bloque cuyo Nº ya se conoce.

b)LLamada del último bloque en el programa completo, entrando el número de bloque seguido de **INTRO.**

(F4) AÑADIR LINEA.

a)Inserción de una instrucción en una secuencia.
Ejemplo:

Insertar una instrucción en la línea Nº5. Posicionar el cursor debajo de la línea 4. A continuación, pulsar la tecla de función (F4) AÑADIR LINEA.

004 AI0.3
= -------- Cursor
005 AI0.5

004 AI0.3
005 ---------Entrada de nuevas instrucciones
por Ej. AI0.4 con /sin comentario
006 AI0.5

b)Cuando se insertan varias instrucciones:
Llamada automática de la siguiente línea con la tecla **INTRO** después de la entrada del comentario; pulsar 2 veces la tecla **INTRO** cuando no hay comentario.

(F5) BUSCAR / REEMPLAZAR.

a)Buscar operandos / sustituir operandos.

b)Buscar bloques / modificar nombres de bloque.

c)Buscar comentarios de bloque /modificar comentarios de bloque.

La función (F5) BUSCAR / REEMPLAZAR hace posible buscar y reemplazar series de caracteres con una longitud máxima de 64 caracteres. En este contexto es preciso tener en cuenta el uso correcto de minúsculas y mayúsculas de los operandos, nombres de bloque y comentarios de bloque que se han de buscar. El espacio en blanco se considera un carácter.
Ejemplo:

“Buscar”

Se busca el nombre del bloque LAB-0:
Entre el texto deseado en mayúsculas: LAB-0.
Responda con (N) NO a la pregunta: “Desea reemplazar el texto”
La operación de búsqueda se realiza de pantalla en pantalla. Responda a la pregunta: **“Desea buscar otras entradas”** con (S) SI. El texto buscado aparece en video inverso.
El mensaje en pantalla: **“Serie de caracteres no encontrada”** significa que el **BLOQUE1** no existe otra vez en el programa.

Observación:

a)Cuando se crean o modifican ficheros de referencias, también pueden buscarse y reemplazarse textos de comentario. (Seleccionar la función: Buscar serie de caracteres). En este caso deben entrarse exactamente los mismos caracteres que llevan los textos que se han de buscar o reemplazar.

b)Si por Ej. se ha de buscar el programa de usuario completo que se ha de entrar:
De bloque (0 a..) INTRO.....) o bien marca con **INTRO y ....a bloque**: confirmar con **INTRO.**

c)La función reemplazar funciona de la misma manera.

d)La función buscar puede encontrarse en el submenú **“Visualización de estados en LDI” del menú “Test / puesta en marcha”**

(F6) COPIAR BLOQUES.

Todos los bloques que tienen el mismo contenido pueden copiarse en el lugar correspondiente del programa con el objeto de racionalizar el trabajo de programación.
La entrada **"Antes del bloque: xxx"** significa que el bloque copiado se ha de insertar antes del bloque Nº xxx.

Observación:

De bloque (...... a ....) o de marca: xxx **a marca:** xxx significa copiar el bloque Nº xxx.
De bloque (.......a ......) o de marca: xxx **a bloque:** yyy significa copiar los bloques Nº xxx a Nº yyy inclusive.

(F7) BORRAR BLOQUES.

Existe la posibilidad de borrar uno o varios bloques que perjudican el funcionamiento general del autómata. Aquí pueden aplicarse las mismas instrucciones que han s ido dadas para la función **"Copiar bloques".**

(F8) BORRAR LINEA EN USO.

Esta función se utiliza para borrar la línea marcada con el cursor. Las restantes líneas se desplazan hacia arriba para que no quede ninguna línea vacía.

(F9) COMENTARIO DE OPERANDO.

Es posible asignar a cada operando un comentario que se puede modificar posteriormente.

Opciones de corrección durante la programación

Estas opciones pueden aplicarse durante la programación como durante la modificación de programas ya existentes. Esto rige también para los ficheros de referencias.

a)Control del cursor.-El cursor puede desplazarse a la posición deseada mediante las teclas de posicionamiento del cursor.

b)Movimiento al inicio del programa: (HOME) o (INICIO).

c)Movimiento al final del programa: (END) o (FIN).

d)Movimiento a la siguiente página cuando el cursor está en la última línea de la página: (Pg Dn) o (Av Pág).

e)Movimiento a la página anterior cuando el cursor está en la primera línea de la página: (Pg Up) o (Re Pág).

MENU PRINCIPAL ---- PROGRAMACIÓN——EDITAR —————— INSERTAR

F1	RETORNO	F4	AÑADIR LINEA	F7	BORRAR BLOQUES
F2	ABRIR BLOQUE	F5	BUSCAR /SUSTITUIR	F8	BORRAR LINEA ACTUAL
F3	BUSCAR BLOQUE	F6	COPIAR BLOQUES	F9	COMENTARIO OPERANDO

Ejemplo:
Modificar I0.2 en AI0.2: Posicionar el cursor debajo de la I, pulsar la tecla (A).

.
.
.
003 I0.2 ------ Cursor
=
.
.
.
003 AI0.2 ------ Cursor

=

f)Sustituir caracteres. Pase del modo Insertar al modo sobreescribir con la tecla **(INSERT).**

MENU PRINCIPAL ---- PROGRAMACIÓN——EDITAR —————— Sustituir

F1	RETORNO	F4	AÑADIR LINEA	F7	BORRAR BLOQUES
F2	ABRIR BLOQUE	F5	BUSCAR /SUSTITUIR	F8	BORRAR LINEA ACTUAL
F3	BUSCAR BLOQUE	F6	COPIAR BLOQUES	F9	COMENTARIO OPERANDO

Se puede sustituir el carácter bajo el cual está posicionado el cursor.

Ejemplo:
Modificar AI0.2 en OI0.2 :
.
.
003 AI0.2 ------Cursor
=
.
.
.
003 OI0.2

=

g)Borrar caracteres. Los caracteres pueden borrarse individualmente de dos maneras diferentes:

1.-Usando la tecla (DEL) para borrar el carácter debajo del cual se encuentra el cursor.

2.-Usando la tecla de RETROCESO para borrar el carácter que se encuentra a la izquierda del cursor.

Ejemplo:

Borrar la letra A de la palabra TESTA.

1.-Posicionar el cursor debajo de la letra A, pulsar la tecla (DEL) o (SUPR).

TESTA **TEST**
= **------ Cursor**

2.-Posicionar el cursor a la derecha de la letra A, pulsar la te cla de **( ) RETROCESO.**

TESTA **TEST**
= **------ Cursor**

MENU PRINCIPAL ---- PROGRAMACIÓN ------------c: ejercici . q3 ------c: ejercici . z3

F1 RETORNO	**F4 SALVAR DATOS**	**F7 SALIDA PROGRAMA FUENTE**
F2 EDITAR PROGRAMA		**F8 SALIDA LISTA DE REFERENCIA**
F3 EDITAR LISTA REF.		**F9 SALIDA FORMATO PRG 3**

Los **ficheros fuente** y **de referencias** que se encuentran en el editor de programas se pierden al desconectar el ordenador personal; por esta razón han de salvarse en un diskette o bien en el disco duro. El display anterior aparece cuando se sale del editor de programas pulsando la tecla de función **(F1) INTRO**. Al pulsar de nuevo **(F1) INTRO**, se visualiza en pantalla lo siguiente:

Se han modificado los siguientes archivos :

Programa fuente c : ejercici.q3

Lista referencia c : ejercici . z3

Desea salvar los archivos modificados ? (S / N)

F1 RETORNO	**F4 SALVAR DATOS**	**F7 SALIDA PROGRAMA FUENTE**
F2 EDITAR PROGRAMA		**F8 SALIDA LISTA DE REFERENCIA**
F3 EDITAR LISTA REF.		**F9 SALIDA FORMATO PRG 3**

Si la pregunta referente a salvar los archivos se responde con **(N) NO**, el programa regresará al menú **"Programación"** y los ficheros creados se borrarán.
Si se responde a la pregunta con **(S) SI**, el programa regresará al menú **"Salvar"** igual que si hubiese seleccionado **(F4) SALVAR PROGRAMAS** en el display anterior o en el que se muestra a continuación.

```
MENU PRINCIPAL -------- PROGRAMACION —— SALVAR ——————————
F1  RETORNO                       F4  SALVAR LISTA REFERENCIA
F2  SALVAR NOMBRE ACTUAL
F3  SALVAR PROGRAMA FUENTE                              F10  AYUDA
```

Con la tecla de función **(F2) SALVAR NOMBRE ANTIGUO** se memorizan los dos ficheros, el **fichero fuente** y el **fichero de referencias**. Esto se visualiza en pantalla como sigue:

Salvar el fichero c: Ejercicio.q3

Salvar el fichero c: Ejercicio.z3

Después de la memorización de los dos ficheros el programa retorna automáticam ente al menú anterior.

h)Salida de un fichero. Los programas almacenados en un diskette o en el disco duro pueden salir como ficheros individuales **(fichero fuente, fichero de referencias)** con diversas opciones.

```
MENU PRINCIPAL ---- PROGRAMACIÓN ------------c: ejercici . q3 ------c:  ejercici . z3

F1  RETORNO              F4  SALVAR DATOS     F7  SALIDA PROGRAMA FUENTE
F2  EDITAR PROGRAMA                           F8  SALIDA LISTA DE REFERENCIA
F3  EDITAR LISTA REF.                         F9  SALIDA FORMATO PRG 3
```

Partiendo del submenú de programación se pulsa la tecla de función **(F7 - F9)** en función del tipo de salida que se desee y aparece en pantalla el siguiente display después de haber indicado desde dónde hasta dónde (especificar Nº de bloque) se desea la salida:

```
MENU PRINCIPAL -------- PROGRAMACION --------SALIDA
F1  RETORNO               F4  DISKETTE / DISCO FIJO
F2  PANTALLA
F3  IMPRESORA                                   F10  AYUDA
```

Ahora se ha de especificar dónde ha de realizarse la salida:

(F2) Pantalla El programa sale en pantalla.

(F3) Impresora El programa se imprime a través de la impresora.

La salida puede interrumpirse pulsando cualquier tecla.
Si se ha pulsado una tecla de función, acto seguido se llevará a cabo la correspondiente función, si ello es posible. Si no se ha pulsado ninguna tecla de función existe la posibilidad de que el programa salga de nuevo pulsando una tecla cualquiera.

(F4) Diskette / Disco duro: El programa sale en Diskette / Disco duro. En este caso se utiliza el mismo formato que el visualizado en pantalla. El fichero se almacena con el nombre "Nombre de fichero.xls" en la unidad de disco en uso, desde donde se había cargado el programa. "Nombre de fic hero", significa, aquí el nombre del programa y "x" el tipo, es decir "q" para fichero fuente, "z" para fichero de referencias.

```
SUCOSOFT   S30

Especifique un nombre para :   Lista de referencia

    Nombre  : Ejercici

    Disco :  c

       Con (F1) hasta (F10) puede interrumpir la  introducción  del nombre

Camino : C:\PS3\U23\AWB
```

Cuando se crea un programa, además de establecer el nombre para el programa de usuario, se asigna automáticamente a l mismo tiempo un nombre para el fichero de referencias. Si se escribe solamente en **LDI** puede escribirse también el comentario de operando que se almacena en el fichero de referencias. De este modo se obtiene un fichero completo de referencias, además de u nas pocas entradas adicionales. También es posible proceder en sentido inverso, es decir, escribir en primer lugar el fichero de referencias y a continuación el fichero de programa. Esto es absolutamente necesario cuando se realiza la programación con oper andos simbólicos y es aconsejable cuando se programa en esquema de contactos **(EDC).**
Existe la posibilidad de crear los datos de referencias por separado para no tener que ir cambiando siempre de menús.
Pulsando la tecla de función **(F5) FICHERO DE REFERENCI AS** en el submenú de programación se inicia la adición del fichero de referencias y aparece en pantalla el display anterior.
Después de entrar el nombre y especificar la unidad de disco se pulsará la tecla **RETORNO** y se visualizará el siguiente menú de selec ción:

MENU PRINCIPAL -------- PROGRAMACION ————————		Ejercici.z3
F1 RETORNO	F4 SALIDA LISTA REFERENCIA	
F2 EDITAR LISTA REF.		
F3 SALVAR DATOS		F10 AYUDA

Para trabajar en el fichero de referencias se pulsará la tecla de función **(F2) FICHERO DE REFERENCIAS** con lo que se visualizará el siguiente display:

Símbolo	Operando	V	Borne	Comentario del operando
S0	I0.0	S	1X0	Pulsador S0 accionado
S1	I0.1	S	1X1	Pulsador S1 accionado
S3	I0.3	S	1X3	Arranque TR0
S4	I0.4	S	1X4	Interrupción TR0
M1	Q0.1	S	2X1	Motor M1 con.
H0	Q0.0	S	2X0	Indicador luminoso S2 con.
S2	TR0	S	1X2	Retardo en el arranque
	KW50			Tiempo transcurrido = S2

MENU PRINCIPAL-----PROGRAMACIÓN-----EDITAR LISTA REF.———---Insertar		
F1 Retorno	F4 Añadir linea	F7 Borrar campo actual
F2 Escribir comentarios	F5 Buscar texto	F8 Borrar linea actual
F3 Cargar comentarios	F6 Ordenar lista	F9 Borrar area

<u>**Explicación de las claves del fichero de referencias:**</u>

Símbolo : Nombre simbólico o código del equipo (8 caracteres).

Operando: I, Q, N y nombres de módulos.

V : Comportamiento del equipo, por ej. contacto de cierre o de apertura (2 caracteres).

Terminal : Bloque de terminales y Nº de terminal u otros datos (12 caracteres).

Comentario: Función del equipo (40 caracteres).

Cuando un símbolo o un operando se indican más de una vez se visualiza el correspondiente mensaje.
Solamente será transferido al programa el primer texto de comentario de cada s ímbolo u operando que esté presente más de una vez en el programa.
Por lo demás, en la creación del fichero de referencias se aplican los mismos criterios generales que para la programación. A continuación se describen las otras funciones existentes en el fichero de referencias.

(F1) RETORNO
Regreso al programa **PROGRAMACION**.

(F2) ESCRIBIR COMENTARIOS.
Los comentarios cargados con **(F3)** se escriben en la línea en uso. Cualquier otro comentario existente se sobreescribirá.

(F3) CARGAR COMENTARIOS EN USO.
Los tres campos **"V"**, "Terminal" y "Comentario de operando" se cargan en una memoria interna para que puedan ser llamados en caso de que se necesiten.

(F4) AÑADIR LINEA.
Con esta función se abre una nueva línea debajo de la línea en uso.

(F5) BUSCAR SERIE DE CARACTERES.
En el fichero de referencias se puede buscar cualquier serie de caracteres con una longitud máxima de 40 caracteres. Una vez entrada una serie de caracteres ésta permanece intacta hasta que se borra con la tecla de retroceso o se sobreescribe automáticamente al entrar el siguiente texto de búsqueda. La operación de búsqueda empieza en la línea que sigue a la línea en uso, continua hasta el final del programa y regresa al principio. Termina en la línea en uso o bien cuando se ha encontrado la serie de caracteres.
La operación de búsqueda puede continuar pulsando la tecla de función **(F5)** y la tecla **RETORNO**. El cursor salta al primer carácter de la serie si la búsqueda ha tenido éxito.

(F6) CLASIFICAR LISTA.
El fichero de referencias puede clasificarse en tres campos diferentes:

1.-Por símbolos.
2.-Por operandos.
3.-Por terminales.

Las líneas que no contienen ningún dato en el criterio de clasificación indicado (Campo)aparecerán más tarde en su secuencia original, al final del fichero de referencias.

(F7)BORRAR CAMPO EN USO.
Aquí se borra el campo en el cual está posicionado el cursor.

(F8)BORRAR LINEA EN USO.
Aquí se borra la línea en la cual está posicionado el cursor.

(F9) BORRAR AREA.
Con la tecla de función **(F9)** puede marcarse un área de la lista de referencias y puede borrarse después. El área marcada no se ve en la pantalla.

SALVAR.

Una vez creado, el fichero de referencias puede salvarse con fines de archivo en una de las unidades de disco, es decir todos los programas modificados s e vuelven a escribir en la unidad de disco especificada al asignar el nombre. En este caso las versiones previas quedan sobreescritas. En la unidad de diskette esto puede evitarse introduciendo otro diskette.
Pulsando la tecla **RETORNO** se regresa al menú de **"PROGRAMACIÓN".**

```
MENU  PRINCIPAL -------- PROGRAMACION ______________________  Ejercici.z3

F1   RETORNO                     F4   SALIDA LISTA REFERENCIA
F2   EDITAR LISTA REF.
F3   SALVAR DATOS                                             F10   AYUDA
```

Pulsando de nuevo la tecla **(F1) RETORNO** se regresa al menú **"SALVAR"**

```
MENU  PRINCIPAL -------- PROGRAMACION ______________________

 F1   Retorno
 F2   Salvar nombre actual
 F3   Salvar nombre nuevo                              F10   Ayuda
```

Con la tecla de función **(F2)SALVAR NOMBRE ANTIGUO**, el fichero de referencias que ha sido modificado durante su edición se vuelve a escribir en la unidad de disco correspondiente con el nombre original.
Con la tecla de función **(F3) SALVAR NOMBRE NUEVO** el fichero de referencias editado puede salvarse con el nuevo nombre. También es posible especificar de nuevo la unidad de disco donde los ficheros han de ser a lmacenados.

Operandos simbólicos:

a)

00000 BLOQUE0	**"Inicio"**	
001	**LI0.0**	**Tecla SO accionada.**
002	**AI0.1**	**Tecla SO accionada.**
003	**OI0.2**	**Interruptor S2 con.**
004	**=Q0.0**	**Indicador luminoso HO con.**
00001 BLOQUE1	**"Fin"**	
001	**EP**	

b)

00000 BLOQUE0	**"Inicio"**	
001	**L'SO**	**Tecla SO accionada.**

002	**A’S1**	**Tecla S1 accionada.**
003	**O’S2**	**Interruptor S2 con.**
004	**=‘HO**	**Indicador luminoso HO con.**
00001 BLOQUE1	**“Fin”**	
001	**EP**	

Junto a la programación normal en lista de instrucciones **LDI** con operandos absolutos **(a)** también hay la posibilidad de crear programas con oparandos simbólicos (**b),** es decir:
Todos los operandos (I, Q, M, etc.)pueden representarse en el programa tanto con los códigos de identificación de los operandos y los parámetros (por ej. I0.0) como en calidad de operandos simbólicos (por ej. SO). Los operandos simbólicos constan de un máximo de 8 caracteres que pueden seleccionarse opcionalmente.
Se enumeran en el fichero de referencias y se han de asignar al operando pertinente, pues de lo contrario no se compilan.
En el fichero de referencias pueden entrarse datos adicionales para V = comportamiento (por ej. C = contacto de cierre), código de identificación de los terminales y comentario de operando. Los operandos simbólicos deben ir precedidos del signo comilla simple (‘) cuando se entran en la sección del programa, a fin de poderlos identificar mejor. El modo de proceder para la programación con operandos simbólicos es por lo demás idéntica a la programación con operandos absolutos.

Ejemplo de un fichero de referencias:

Símbolo	Operando	V	Terminal	Comentario del operando
S0	I0.0	S	1 x 1	Pulsador S0 accionad o
S1	I0.1	S	1 x 2	Pulsador S1 accionado
S2	I0.2	S	1 x 3	Interruptor S2 con
HO	Q0.0	-	2 x 1	Indicador lum. HO con

<u>Compilador:</u>

El programa del usuario creado en el editor de programas ha de ser traducido a un lenguaje comprensible para el PLC.
Esta traducción es llevada a cabo por el compilador y siempre es necesaria después de crear un nuevo programa y después de hacer modificaciones o de completar un programa.
El compilador se selecciona pulsando la tecla de función **(F6) COMPILADOR** en el submenú de programación.

```
                    SUCOSOFT  S30

        PROGRAMACIÓN

        F1   MENU PRINCIPAL
        F2   PROGRAMACIÓN  LDI
        F3   PROGRAMACIÓN  EDC
        F4   PROGRAMACIÓN  EBF
        F5   LISTA DE REFERENCIA
        F6   COMPILADOR

        F10   AYUDA

        Elija por favor

Camino : C:\PS3\U23\AWB
```

Una vez seleccionado el compilador con la tecla de función **(F6)** se entran los nombres para el fichero fuente y para el fichero de referencias con su correspondiente memoria (Unidad de disco a: o bien c:), tal como se ha descrito en la sección "Establecimiento del nombre del programa del usuario".

El fichero fuente y el fichero de referencias no han de tener necesariamente el mismo nombre cuando por ej. hay varios programas de usuario que tienen acceso a un fichero de referencias.

Los nombres de los ficheros y las especificaciones de las unidades de disco se entran y las entradas se completan pulsando la tecla **RETORNO**. A continuación se inicia el proceso de compilación (traducción).

Esto se visualiza en la pantalla con diferentes mensajes breves que aparecen uno a continuación de otro.

Si el compilador detecta un error en el programa, por ej. un cambio del tipo de dato dentro de la misma secuencia, queda reflejado en el texto legible con la situación exacta del error. En tal caso no se genera ningún código PS, lo cual quiere decir que no se realiza la traducción al lenguaje de máquina.

```
c: ejercici . q3     Bloque  0   Linea  4 : ---------
              Imposible cambiar el tipo de dato dentro de una secuencia?


Se han encontrado  1  fallo (s)
No se ha creado un archivo compilado?

Seguir con (INTRO)
```

Después de haber regresado al editor de programas y de haber corregido el error, el programa puede compilarse de nuevo una vez se ha salvado después de haberse corregido.

Cuando el programa no tiene ningún error se genera un código PS, es decir, se genera un programa en lenguaje de máquina. Este código se almacena en el archivo creado automáticamente xxx.p3.

En pantalla aparecerá el siguiente mensaje:

Se han detectado 0 fallas.
Han sido programadas 3 instrucciones LDI.
Memoria requerida = 8 Bytes
Memoria disponible = 3672 Bytes
Seguir con **RETORNO.**

Ahora el programa puede ser transferido al PLC.

TRANSFERENCIA UNIDAD DE DISCO -------- PS

SUCOSOFT S30

MENU PRINCIPAL

F1 PROGRAMACIÓN
F2 TEST/PUESTA EN MARCHA
F3 DOCUMENTACIÓN
F4 TRANSFERENCIA
F5 COMPARACIÓN
F6 ADMINISTRACIÓN DE ARCHIVOS
F7 ELECCIÓN DEL SISTEMA
F8 MS - DOS

F10 AYUDA

Elija por favor

Camino : C:\PS3\U23\AWB

Llamada de transferencia del programa con la tecla de función **(F4) TRANSFERENCIA** en el menú principal.
Solamente son transferidos los programas que están compilados (traducidos) sin ningún error. Son posibles varias direcciones de transferencia.

SUCOSOFT

TRANSFERENCIA S3

F1 MENU PRINCIPAL
F2 UNIDAD -------PS
F3 PS -------UNIDAD
F4 DESCOMPILADOR

F10 AYUDA

Elija por favor?

Camino : C:\PS3\U23\AWB

Para transferir un programa completo del PC al autómata se ha de pulsar la tecla de función **(F2) UNIDAD DE DISCO ----- PS** y a continuación aparecerá en pantalla el siguiente display:

```
SUCOSOFT

TRANSFERENCIA

DISCO --------------------------PS

     Se transfieren solo programas ejecutables (compilados)

     Nombre  : ejercici
     Disco  :
     Disco actual : c

     CON (F1)  HASTA (F10) RETORNO AL MENU


Camino  :  C:\PS3\U23\AWB
```

Entre el nombre del programa que ha de ser transferido y la unidad de disco **"C",** desde la cual se ha de realizar la transferencia. Complete la entrada pulsando la tecla **RETORNO**. La operación de transferencia queda indicada por el siguiente menú:

```
SUCOSOFT

TRANSFERENCIA

DISCO --------------------------PS



     Nombre  : ejercici
     Disco  :  c



        Espere, por favor

Camino  :  C:\PS3\U23\AWB
```

Una vez concluida la transferencia se regresa al submenú de transferencia.
Cuando el **PLC** no está conectado o cuando el cable de transferencia está dañado aparece en pantalla el siguiente mensaje:

"Error de transmisión de datos"

También es posible realizar del mismo modo la transferencia en otra dirección
Submenú test / puesta en marcha.

SUCOSOFT S 30

TEST / PUESTA EN MARCHA

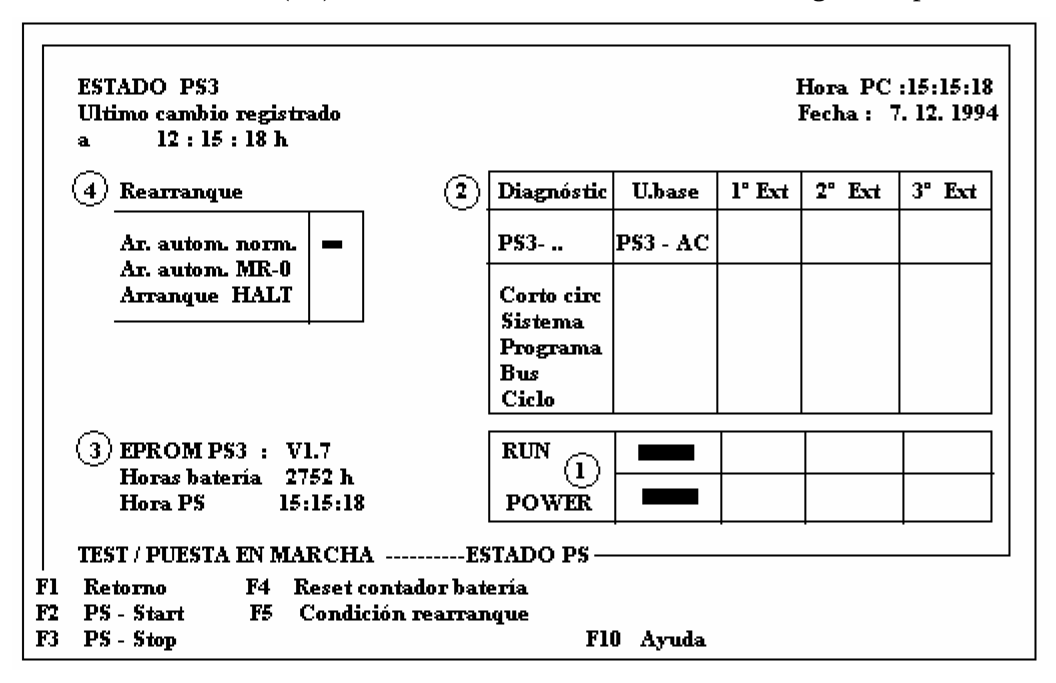
F1 MENU PRINCIPAL
F2 ESTADO DEL PS
F3 TEST DEL HARDWARE
F4 INDICACIÓN ESTADO LDI
F5 INDICACIÓN ESTADO EDC
F6 INDICACION ESTADO EBF
F7 FECHA / HORA

F10 AYUDA

Elija, por favor

Camino : C :\ps3\U23\AWB

La capacidad que poseen los autómatas programables para su interconexión en configuraciones descentralizadas de hasta 600 m longitud, así como la habilidad para crear complejas estructuras de programas de usuario hacen aconsejable llevar a cabo bastantes test del autómata y del programa antes o bien durante su puesta en marcha.
El submenú **TEST/PUESTA EN MARCHA** ofrece numerosas opciones para ello.
Este menú se selecciona pulsando la tecla de función **(F2) TEST/PUESTA EN MARCHA**, con lo que aparecerá el menú mostrado en la figu ra arriba.
Las funciones individuales solo podrán llevarse a cabo cuando el autómata esté conectado. Si aparece en pantalla el mensaje **"Error de comunicación"** significa o bien que el autómata no está respondiendo (por ej. que no hay tensión) o bien que el cable de datos no está conectado o tiene falla.
Pulsando la tecla de función (F2) **ESTADO DEL PS** se visualiza la siguiente pantalla:



Esta visualización ofrece una vista general de una interconexión de autómatas **PS** con un **PS3** como aparato base y tres **PS3** como esclavos de extensión.

Estados del PS.-

Funciones básicas 1 :

RUN

- Display activado con (F2) — PS3 en estado RUN
- Desactivado con (F3) — PS3 en estado HALT
- El display del aparato base parpadea: — Aviso de error, ver diagnóstico

POWER Especifica el aparato base y todos los esclavos direccionados en el programa.

Diagnóstico 2 :

CORTOCIRCUITO Display de cortocircuito en tipo de PS3 con control de cortocircuito (PS3 DC).

SISTEMA Error de hardware en el aparato base.

PROGRAMA Aparato base sin programa.

BUS Señaliza error de transferencia entre el aparato base los racks de extensión (esclavos). Los racks de exten - sión que señalizan **RUN** no son direccionables desde el aparato base y ponen a cero automáticamente sus salidas.

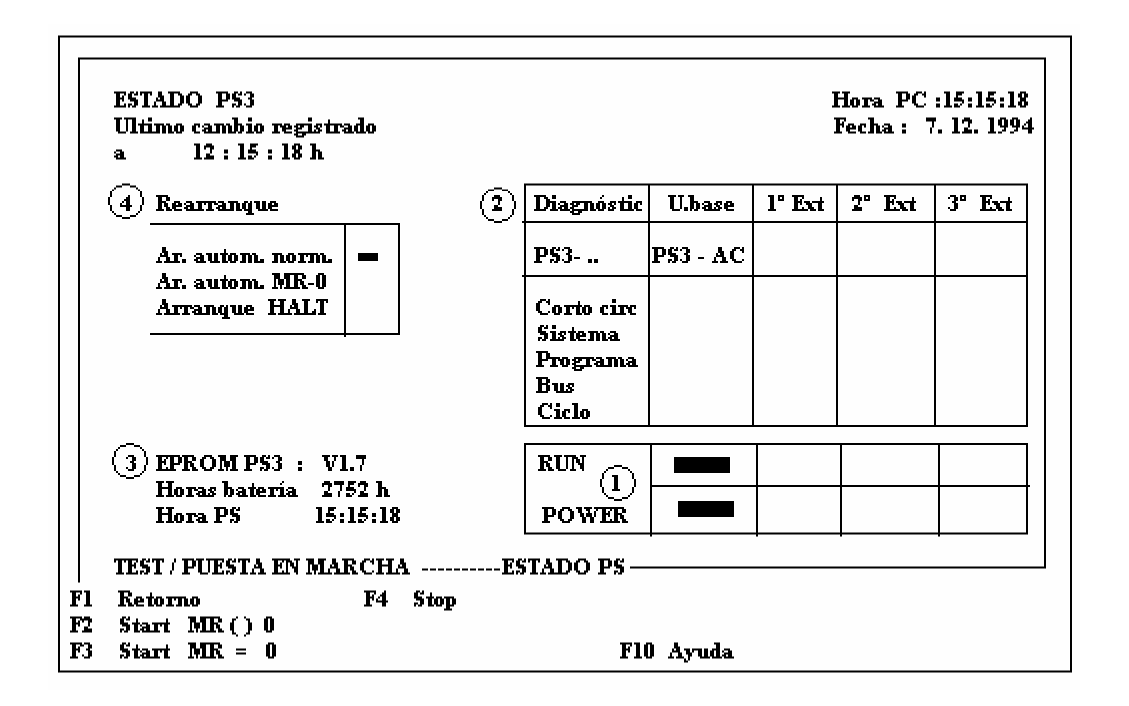
CICLO Control de ciclo. Se ha sobrepasado el tiempo máxi - mo del programa de usuario..

Horas de batería 3 :

Visualización del contador de horas de la batería con posibilidad de puesta a **“0”.**

(F4) HORAS DE BATERIA / RESET.

El contador únicamente está activo cuando el aparato está desconectado de l a red y las baterías están colocadas.



Condición de rearranque 4 :

Esta función permite preestablecer por software, a través del aparato de programación **(PC**), la condición de rearranque, es decir, el comportamiento del autómata despu és de un fallo de la tensión, cuando ésta ya se haya restablecido.
La selección se realiza pulsando la tecla de función **(F5)** Condición de **REARRANQUE,** con lo que se visualizará el display anterior.
Hay tres opciones:

(F2) AUTOARRANQUE: Arranque automático del PS3 al conectar la tensión.

normal: Los Merkers se ponen a cero

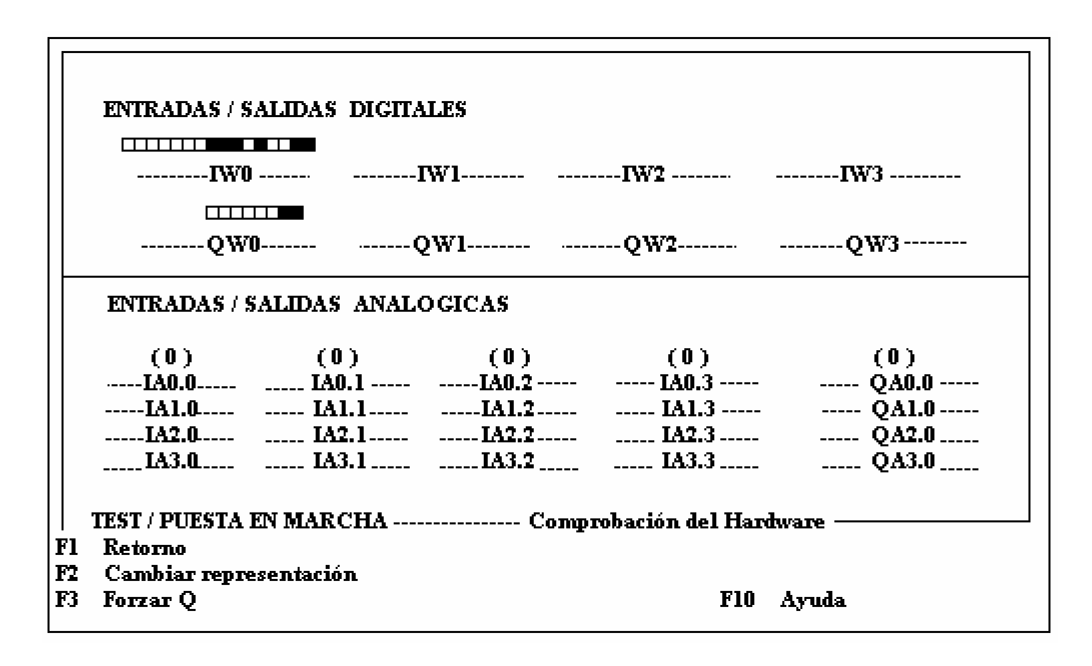
MR = <>0: Los Merkers remenentes y módulos conservan su informa - ción.

(F3) AUTOARRANQUE: Arranque automático del PS3 al conectar la tensión.

MR = 0: Merkers, Merkers remanentes y módulos se ponen a **"0".**

(F4) STOP: Ningún arranque automático del PS3. Arranque a través del

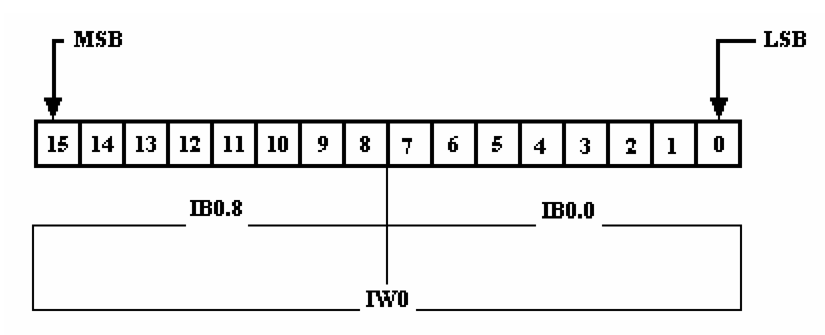
PC. Los Merkers se comportan como en el **"autoarranque normal"**



Pulsando la tecla de función **(F3) TEST DE HARDWARE** se visualiza una representación de los estados de todas las entradas y salidas digitales y analógicas del PS3.
El display de las entradas y salidas digitales se hace en representación binaria, es decir se visualizan palabras de entrada y salida **(por ej. IW0, QW0)** con sus patrones bit **(0, 1).**
El display de las entradas y salidas analógicas puede hacerse tanto en representación binaria como decimal, es decir con sus valores numéricos **(0 - 255).** Con la tecla de función **(F2) CAMBIAR REPRESENTACION** se cambia de la forma de representación decimal a binaria y viceversa.
Las entradas y salidas del aparato base conectado están representadas por campos marcados. Ver display anterior.

Campo vacío **=** Pantalla oscura = 0
Campo lleno **=** Pantalla clara = 1

Display de las entradas y salidas:
Por ej. **I .....0 - 1.....15**



Visualización de estados en LDI.-

Además de la visualización de los estados del **PS** y del **test de hardware** hay otra posibilidad para simplificar el test y la puesta en marcha de un autómata:
La visualización de estados es factible en la programación en lista de instrucciones **LDI,** esquema de contactos **EDC** y en esquema de funciones **EDF** y muestra en la pantalla del **PC** los estados de los operandos **(I, Q, M)** de un programa en el autómata.
Pulsando la tecla de función **(F4) VISUALIZACION DE ESTADOS EN LDI** se selecciona este menú. A continuación se entran el nombre del programa en el autómata y el nombre del fichero de referencias así como las correspondientes unidades de disco, con lo que el programa se visualiza en la pantalla del PC. (ver el siguiente menú).

Observaciones importantes:
Después de seleccionar la función **VISUALIZACION DE ESTADOS** con la tecla de función **(F4)** pueden aparecer en pantalla los siguientes mensajes:

1.-**Error de transmisión de datos! Intentar de nuevo? (S/N).**
Este aviso significa que el autómata no tiene tensión o bien que el cable de datos se ha interrumpido.

2.-**El autómata no está en RUN! Ha de efectuarse un arranque del autómata? (S/N).**
El autómata está bajo tensión pero el programa no se está ejecutando.

3.-**No hay ningún programa en el autómata. Por favor,transfiera programa! (S/N).**
En este caso ha de realizarse una transferencia de programa del **PC al PLC** y a continuación ha de seleccionarse de nuevo la **VISUALIZACION DE ESTADOS EN LDI** pulsando la tecla de función **(F4).**

```
00000  BLOQUE0          "Inicio

001               1     LI0.0               Pulsador S0 accionado
002               1     AI0.1               Pulsador S1 accionado
003               0     OI0.2               Interruptor S2 con.
004               1     =Q0.0               Indicador luminoso S2 con.
00001  BLOQUE1          "Tiempo
001                     TR0                 Retardo en el arranque
002               0     ( ) S: I0.3         Arranque TR0
003               0     ( ) STOP : I0.4     Interrupción TR0
004                     (w) I : KW50
005               0     ( ) EQ : Q0.1       Motor M1 con.
00002  BLOQUE2           "Fin
001                      EP

                                            Transmisión de datos activa
-MENU PRINCIPAL ----VISUALIZACIÓN DE ESTADOS ---------Decimal
F1   Retorno             F4  Indicación estado SI    F7  Representar en dec. / hex.
F2   Seleccionar bloque  F5  Indicación estado NO    F8  Representar en + / -
F3   Buscar texto                                    F9  Representar en binario
```

Para activar el registro de estados pulse la tecla de función **(F4) REGISTRO DE ESTADOS CON..**
Los estados de los operandos que se encuentran en la pantalla se visualizan delante del comienzo de la correspondiente línea del programa.
Las barras luminosas de distintas longitudes se usan para distinguir mejor los diferentes tipos de datos **(Bit, Byte, Word).** Pulsando una tecla cualquiera el registro no desaparece (El display queda "congelado"). El mensaje **"Transmisión de datos activa"** que parpadea durante la transmisión cambia a **"Transmisión de datos no activa".**

"1" En secuencias tipo Bit significa: Contacto cerrado.

"0" En secuencia tipo Bit significa : Contacto abierto.

Los operandos tipo **Byte** y tipo **Word** pueden representarse con su valor decimal o hexadecimal.

(F5)REGISTROS DE ESTADO DESCON.
Pulsando esta tecla de función se puede desconectar el registro de estados. Los valores registrados por último y mostrados en la pantalla se conservan. El display desaparece.

(F7)REPRESENTACION DEC./HEX.
Pulsando esta tecla de función aparecen en forma hexadecimal los estados de los operandos y de los módulos del sistema, forma de representación que requieren algunas aplicaciones. Por ej. los estados de contadores o de los registros de desplazamiento pueden controlarse más fácilmente que en representación decimal. La representación hexadecimal está indicada por el carácter # a la derecha del valor del estado del operando correspondiente.

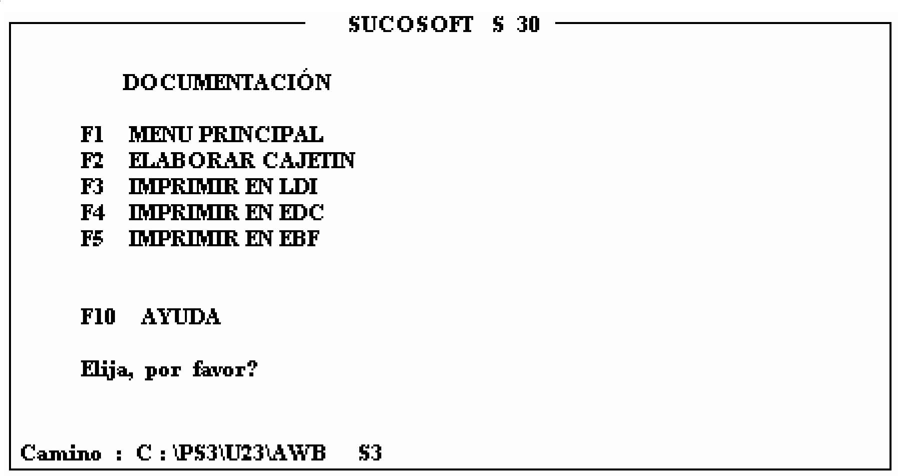
(F8)REPRESENTACIÓN +/-.
Selección entre representación decimal con signo o sin signo. La tecla de función **(F8)** solamente se activa en el caso de representación decimal.

(F9)REPRESENTACIÓN BINARIA.
Los estados de los operandos se visualizan en formato binario a la derecha de las instrucciones **LDI.** A la izquierda de las instrucciones **LDI** éstos se visualizan en formato en formato hexadecimal. En la representación binaria se inserta cada cuatro dígitos un espacio en blanco para facilitar la legibilidad. El bit de mayor peso está situado a la izquierda.

(F2)SELECCIONAR BLOQUE.
Después de entrar un número de bloque o una etiqueta de bloque el display muestra el punto requerido en el programa y el cursor está posicionado al principio del bloque seleccionado. La entrada de los números de bloque puede realizarse sin ceros a la izquierda. Pulsando la tecla **(F1) RETORNO** se selecciona el último bloque en el programa. Pulsando una tecla de función o una tecla de posicionamiento del cursor la función queda interrumpida. La función asignada a la tecla pulsada es ejecutada directamente.

(F3)BUSCAR SERIE DE CARACTERES.
Es posible buscar cualquier serie de caracteres con una longitud máxima de 40 caracteres. Entre la serie de caracteres deseada (por ej. para un número de blo que, etiqueta de bloque, módulo de sistema u operando), pulse a continuación **RETORNO** y el cursor se colocará sobre las distintas posiciones de las series de caracteres existentes en el programa.

(F1) RETORNO. Después de pulsar esta tecla el programa regre sa al menú Test y puesta en marcha.



SUCOSOFT S 30

DOCUMENTACIÓN

F1 MENU PRINCIPAL
F2 ELABORAR CAJETIN
F3 IMPRIMIR EN LDI
F4 IMPRIMIR EN EDC
F5 IMPRIMIR EN EBF

F10 AYUDA

Elija, por favor?

Camino : C : \PS3\U23\AWB S3

En el submenú programación también es posible imprimir programas en lista de instrucciones **LDI**, Esquema de contactos **EDC** y esquema de bloques funcionales **EDF**, así como también imprimir el fichero de referen cias en forma de listados.
En los diferentes tipos de programación, por ej. en **LDI,** hay distintas posibilidades para una salida de fichero con:

(F2) PANTALLA.
(F4) DISKETTE O DISCO DURO.
(F3) IMPRESORA.

SUCOSOFT S 30

ADMINISTRACIÓN DE ARCHIVOS S3

F1 MENU PRINCIPAL
F2 INDICE DE ARCHIVOS
F3 BORRAR ARCHIVOS
F4 COPIAR ARCHIVOS
F5 UNIR ARCHIVOS
F6 COPIAR DISKETTES
F7 DAR FORMATO A DISKETTES
F8 ADMINISTRACIÓN DE DIRECTORIOS

F10 AYUDA

Elija, por favor?

Camino : C : \PS3\U23\AWB

Cuando se han creado varios programas y éstos se han almacenado en un diskette o en el disco duro, pronto se plantea el problema de ordenar **(administrar)** sistemáticamente estos programas para que más tarde puedan encontrarse fácilmente. Esto, a su vez, facilita la posterior edición o el ulterior procesamiento de los ficheros individuales y, por otra parte, permite a terceros una sinopsis general.
Esta posibilidad se tiene pulsando la tecla de función (F6) en el menú general, con lo que se entra en el submenú **ADMINISTRACIÓN DE ARCHIVOS.**
La función de administración de archivos sólo puede utilizarse con archivos de usuario.
Hay distintos tipos de ficheros de usuario que se identifican por sus extensiones, las cuales están separadas del nombre del fichero por el carácter **" . ".**
Los caracteres "   " de la extensión se reemplazan por el tipo de PS utilizado.

Nombre de fichero . q = Fichero fuente.
Nombre de fichero . z = Fichero de referencias.
Nombre de fichero . p = Fichero de programa.

Ejemplo:

Ejercicio . q3

.... Especificación del tipo de PLC: PS3

..........................Fichero fuente

...........................Separación entre el nombre del fichero y la extensión.

...................................Nombre del fichero.

SUCOSOFT S 30

Indice de archivos

Unidad : c

CON (F1) HASTA (F10) RETORNO AL MENU

Camino : C: \PS3\U23\AWB

Después de pulsar la tecla de función **(F2) ADMINISTRACIÓN DE ARCHIVOS** en el submenú administración de archivos, se le pedirá que especifique la unidad de disco donde se encuentran los archivos que usted desea.

Después de entrar el nombre de la unidad de disco (por ej. **“a”)** y de pulsar la tecla **( )** **RETORNO** se visualizará en pantalla el directorio.

El soporte de datos en la unidad de disco A no tiene nombre

Directorio de A:

EJERCICI	Q3	256	12 - 07 - 94	2:02p
EJERCICI	P3	79	12 - 07 - 94	2:44p
EJERCICI	Q3	256	12 - 07 - 94	2:44p
EJERCICI	Z3	552	12 - 07 - 94	2:44p
ESPA	P3	79	12 - 07 - 94	2:44p
ESPA	Q3	256	12 - 07 - 94	2:44p
ESPA	Z3	552	12 - 07 - 94	2:43p

7 Datei (en) 2.030 Byte

Pulse cualquier tecla, por favor. . .

Si después de esto se presiona la tecla **ENTER** se pasa a la siguiente página de pantalla o bien se regresa al menú superior **(Administración de archivos).**

SUCOSOFT S 30

Borrar archivos

1 Nombre : ejercici

2 Disco : a

3 Unidad de disco en uso : c

4 ejercici.q3 Programa fuente (S / N) :
ejercicoi.z3 Lista referencia (S / N) :
ejercici.p3 Programa compil. (S /N) :

Con F1 hasta F10 puede interrumpir la introducción del nombre

Los archivos que ya no se necesitan pueden borrarse del directorio.
Después de pulsar la tecla de función **(F3) BORRAR ARCHIVOS** en el submenú administración de archivos, se entra el nombre del archivo que se ha de borrar y la unidad de disco en la que se encuentra.
El display: Nombre en uso o unidad de disco en uso indica el nombre y la unidad de disco en el que se encuentra. Si estas entradas han de quedar memorizadas pulse la tecla **RETORNO** pues de lo contrario se ha de hacer una nueva entrada.

Ejemplo anterior:

Unidad de disco: a **Unidad de disco en uso**: c

Los ficheros que se han de borrar aparecen ahora en pantalla. **Pulsando (S) SI o (N) NO** se indica si han de borrarse todos los archivos o solamente algunos.
A continuación el programa regresa al menú anterior.

COPIAR FICHEROS:

```
                         SUCOSOFT S 30

     Copiar archivos

     Fuente Nombre : ejercici          DestiNombre : ejercici

     Disco  : c                        Disco  : a

       ejercici.q3      Programa fuente      (S / N) :
       ejercicoi.z3  Lista referencia        (S / N) :
       ejercici.p3    Programa compil.       (S /N ) :


        Con F1  hasta  F10 puede interrumpir la introducción del nombre
```

La tecla de función **(F4) COPIAR FICHEROS** ofrece la posibilidad de copiar ficheros de programa de unidad de disco a unidad de disco (Por ej. del disco duro **"C"** a la unidad de disco **"A").**
El nombre fuente **(Ejercicio)** con la especificación de la unidad de disco © es el programa que ha de ser copiado.
El nombre destino **(Ejercicio)** con la especificación de la unidad de disco (a) es el programa que ha de copiarse.
El nombre destino puede ser el mismo que el nombre fuente o puede ser diferente.
Si un programa ha de ser copiado en la misma unidad de disco en la que se encuentra el fichero fuente debe asignarse otro nombre puesto que de otro modo el programa se sobreescribiría a sí mismo.
Una vez se han entrado el nombre y la unidad de disco se visualizan en pantalla los ficheros. Pulsando **(S)** SI o **(N)** NO se determina qué fichero o ficheros deben ser copiados.

Copiar ficheros:

Si se especifican los nombres de programa o unidades de disco que no existen aparecerá un aviso de error. Una vez que se ha realizado la entrada correctamente y la operación de copia se ha llevado a cabo el programa regresa el menú previo.

SUCOSOFT S 30

Unir dos archivos	
Archivo 1 = archivo destino	Archivo 2
Nombre : ejercici	Nombre : ejercici
Disco : a	Disco : c
a: ejercici.q3 Programa fuente	c: ejercici.q3 Programa fuente (S/N) :
a: ejercici.z3 Lista referencia	c : ejercici.z3 Lista referen. (S/N) :

Con F1 hasta F10 puede interrumpir la introducción del nombre

Si Ud. por ej. desea volver a utilizar secciones del programa o bien diferentes programas para configurar un nuevo programa, es posible fusionar ficheros individuales pulsando la tecla de función **(F5) FUSIONAR FICHEROS.**
Se seleccionan el nombre y la unidad de disco de los ficheros que han de ser fusionados, por ej. **Ejercicio.q3 con Ejercicio.q3.** Una vez concluida esta operación el programa regresa al menú Administración de archivos.

<u>Tenga en cuenta lo siguiente:</u>
El nuevo programa (diferente) está formado por la combinación del fichero 1 = Fichero destino 1 y 2.

Ejemplo:

00000	**S0**	**“Ejercicio1**	**Fichero 1**
001		**LI0.1**	**(antes de la fusión)**
002		**=Q0.1**	
003		**EP**	

00000	**S0**	**“Ejercicio**	**Fichero 2**
001		**LI0.1**	
002		**=Q0.0**	
003		**EP**	

00000	**S0**	**“Ejercicio1**	**Fichero 1**
001		**LI0.1**	**(después de la fusión)**
002		**=Q0.1**	
003		**EP**	
00001	**S0**	**“Ejercicio**	
001		**LI0.0**	
002		**=Q0.0**	
003		**EP**	

Dado que la operación de fusión es una combinación de ficheros completos se copia también la instrucción **EP (End of Program).**
El nuevo fichero 1 no se podrá ejecutar hasta que no se haya borrado la instrucción **EP** puesto que de otro modo el compilador genera un aviso de error y, el fichero no se podrá compilar.
Para evitar confusiones se recomienda dar otro nombre al fichero 1 después de la fusión.

<u>Copiar diskette:</u>

```
                         SUCOSOFT S 30

    Duplicar diskettes

    ATENCIÓN:
    La utilización del sofware SUCOSOFT S 30 está limitada por
    los derechos del autor. El permiso de uso queda reglamentado
    en el CONTRATO DE UTILIZACION.

    POR FAVOR, TENGA EN CUENTA:
    Los diskettes SUCOSOFT S 30 se pueden usar solamente para
    el uso propio?

    Unidad fuente : a                           Unidad destino : a

            CON (F1) HASTA (F10) RETORNO AL MENU

Camino : C\PS3\U23\AWB
```

Cuando el contenido completo de un diskette ha de copiarse a otro diskette no es necesario copiar cada fichero por separado sino que, esta operación puede simplificarse pulsando la tecla de función **(F6) COPIAR DISKETTE.**
Después de haber seleccionado esta función aparece en pantalla el menú anterior que hace referencia a la función de copiar el **SUCOsoft S 30.**
Entre los datos correspondientes a la unidad de disco fuente y destino y complete las entradas pulsando **RETORNO** respectivamente. A continuación aparece en pantalla la siguiente sucesión de mensajes que corresponden a la operación.

1.- Inserte el diskette fuente en la unidad de disco A:
A continuación pulse cualquier tecla.....

2.-40 tracks
Se están copiando 9 sectores/tracks, 2 caras.

3.-Insertar el diskette destino en la unidad de disco A:
A continuación pulse cualquier tecla....

4.-Operación de copia concluida.
Hacer otras copias ? (S/N).

Ud. puede especificar si desea continuar copiando éste u otros diskettes o no pulsando **(S) SI o (N) NO**. Si se pulsa (N) se regresa al menú.

Nota:

El diskette destino es formateado durante la operación de copia, es decir, el contenido previo se borra.

Para esta operación pueden utilizarse diskette usados o diskettes completamente nuevos sin tener que borrarlos o formatearlos previamente.

Formatear diskette:

```
SUCOSOFT  S 30

Dar formato a diskettes

Unidad : a

CON (F1) HASTA (F10) RETORNO AL MENU
```

Si al seleccionar la función **(F6) COPIAR DISKETTE** el programa copia por completo el contenido de un diskette a otro nuevo y al mismo tiempo lo formatea automáticamente, esto no sucede con ninguna otra operación de copia. En estos casos es necesario formatear previamente el diskette nuevo.

El formateado prepara el diskette para llevar los datos, es decir el diskette es dividido en sectores y tracks y al mismo tiempo se verifica si hay algún tracks defectuoso y se crea un directorio al cual se puede asignar un nombre.

Para formatear un diskette pulse la tecla de función **(F7) FORMATEAR DISKETTE** y especifique la unidad de disco en la que se ha de realizar esta operación. Pulse a continuación la tecla **RETORNO** y aparecerá en pantalla la siguiente secuencia de mensajes:

-Inserte nuevo diskette en la unidad de disco A:
y pulse cualquier tecla.

-Formateando......el diskette ha sido formateado.
Nombre del soporte de datos (11 caracteres, pulse retorno si no se requiere)?

-Memoria total del diskette 362496 Bytes.

Memoria disponible en el diskette / disco duro
362496 Bytes.
Desea formatear otro diskette? (S/N).

Pulsando **(S) SI o (N) NO** se determina si Ud. desea repetir la operación o regresar al menú.